Synchronous Gestures in Multi-Display Environments

Gonzalo Ramos

University of Toronto, Live Labs

Kenneth Hinckley

Microsoft Research

Andy Wilson

Microsoft Research

Raman Sarin

Microsoft Research

RUNNING HEAD:

SYNCH. GESTURES IN DISPLAY ENVIRONMENTS

Corresponding Author's Contact Information:

Gonzalo Ramos

136 – 102nd Ave SE, #226

Bellevue, WA, 98004,

USA

(425)445-3724

gonzalo@microsoft.com

Brief Authors' Biographies:

Gonzalo Ramos received his Honors Bachelors in Computer Science from the University of Buenos Aires where he worked on image compression and wavelets. He later obtained his M.Sc. in Computer Science at the University of Toronto, focusing on numerical analysis and scientific visualization issues. He completed his doctoral studies in Computer Science at the University of Toronto doing research in Human-Computer Interaction. Currently he is a scientist at Microsoft's Live Labs.

Kenneth Hinckley is a research scientist at Microsoft Research. Ken's research extends the expressiveness and richness of user interfaces by designing novel input technologies and techniques. He attacks his research from a systems perspective, by designing and prototyping advanced functionality, as well as via user studies of novel human-computer interactions and quantitative analysis of user performance in experimental tasks. He holds a Ph.D. in Computer Science from the University of Virginia, where he studied with Randy Pausch. This article is dedicated to Randy's heroic battle against pancreatic cancer.

Andy Wilson is a member of the Adaptive Systems and Interaction group at Microsoft Research. His current areas of interest include applying sensing techniques to enable new styles of human-computer interaction, as well as machine learning, gesture-based interfaces, inertial sensing and display technologies. Before joining Microsoft, Andy obtained his B.A. at Cornell University, and M.S. and Ph.D. at the MIT Media Laboratory.

Raman Sarin is a Research Software Design Engineer at Microsoft Research. Raman's current interests include search user interfaces, mobile devices, special-purpose devices, and pen-operated devices, with an emphasis on the realization of research concepts in pragmatic systems that people can use. He holds a B.S. from Rensselaer Polytechnic Institute (RPI).

See hci-journal.com/editorial/final-guidelines.html for explanations of the various parts of an article in *HCI* format.

ABSTRACT

Synchronous gestures are patterns of sensed user or users' activity, spanning a distributed system that take on a new meaning when they occur together in time. Synchronous gestures draw inspiration from real-world social rituals such as toasting by tapping two drinking glasses together. In this paper, we explore several interactions based on synchronous gestures, including bumping devices together, drawing corresponding pen gestures on touch-sensitive displays, simultaneously pressing a button on multiple smart-phones, or placing one or more devices on the sensing surface of a tabletop computer. These interactions focus on wireless composition of physically co-located devices, where users perceive one another and coordinate their actions through social protocol. We demonstrate how synchronous gestures may be phrased together with surrounding interactions. Such *connection-action phrases* afford a rich syntax of cross-device commands, operands, and one-to-one or one-to-many associations with a flexible physical arrangement of devices.

Synchronous gestures enable co-located users to combine multiple devices into a heterogeneous display environment, where the users may establish a transient network connection with other select co-located users to facilitate the pooling of input capabilities, display resources, and the digital contents of each device. For example, participants at a meeting may bring mobile devices including tablet computers, personal digital assistants (PDA's), and smart-phones, and the meeting room infrastructure may include fixed interactive displays, such as a tabletop computer. Our techniques facilitate creation of an ad-hoc display environment for tasks such as viewing a large document across multiple devices, presenting information to another user, or offering files to others. The

interactions necessary to establish such ad-hoc display environments must be rapid and minimally demanding of attention: during face-to-face communication, a pause of even five seconds is socially awkward and disrupts collaboration.

Current devices may associate using a direct transport such as Infrared Data Association (IRDA) ports, or the emerging Near-Field Communication (NFC) standard. However, such transports can only support one-to-one associations between devices, and require close physical proximity as well as a specific relative orientation in order to connect the devices (e.g., the devices may be linked when touching head-to-head, but not side-to-side). By contrast, sociology research in proxemics (the study of how people use the "personal space" surrounding their bodies) demonstrates that people carefully select physical distance as well as relative body orientation to suit the task, mood, and social relationship with other persons. Wireless networking can free device-to-device connections from the limitations of direct transports, but results in a potentially large number of candidate devices. *Synchronous gestures* address these problems by allowing users to express naturally a spontaneous wireless connection between specific proximal (collocated) interactive displays.

CONTENTS

1. INTRODUCTION

The world of technology is a rapidly evolving ecology of devices. This ecology of ubiquitous devices is changing how we employ computing (Weiser, 1991). It includes tabletop and wall displays, tablet computers, handheld devices, cell phones, cameras, music players, wristwatches, and ear buds. Wireless networks have the potential to unite these ubiquitous computational pieces into temporary composite devices that suit a particular task, or that serve a brief collaborative encounter between users, and then dissipate when a task is complete. To realize this potential, users need interaction techniques to marshal spontaneous combinations of the desired devices, input capabilities, and display resources.

For example, people at a meeting could connect their mobile devices to one another or to existing infrastructure, such as an interactive tabletop computer, to form an ad-hoc heterogeneous multi-display environment. Such an environment facilitates collaborative sharing of files and resources with other devices of the user's choosing, and enables new functionality emerging from the combined capabilities of multiple devices. The demands of casual gatherings of heterogeneous multi-device environments differ from traditional single-user desktop computer scenarios in that input, display, and device resources are transient and may fluctuate in their roles with respect to one another. Because of their ad-hoc nature, such multi-device environments cannot *a priori* determine the participating devices' position, orientation, availability throughout time, or the role each device is to play in a given user task. These are all subject to a user's changing whims as he or she collaborates with multiple persons, or with a single person on multiple aspects of a collaborative task.

A dynamic multi-device environment must respect the underlying social fabric of collaborative interaction by providing flexible means for co-located users to work together across a range of physical proximity and relative body orientation. For example, users may wish to (1) view content side-by-side with a co-worker on a shared display, (2) negotiate terms in a face-to-face meeting with a rival seated at the opposite side of a table, or (3) share information with some colleagues on the other side of a crowded meeting room – but not with rivals at the meeting. A technique that requires the devices to rest in close physical proximity at a head-to-head relative orientation might support scenario (2) – provided the table is narrow enough to reach across – but cannot effectively support scenarios (1) or (3). Yet the emerging Near Field Communication (NFC) standard (http://en.wikipedia.org/wiki/Near_Field_Communication), for example, suffers exactly these limitations. (NFC is a low-bandwidth magnetic field induction technique, with a communication range limited to about 20 cm).

The connection mechanism for a dynamic multi-device environment should operate within the context of the users' task workflow, rather than addressing the connection step in isolation. We present techniques that phrase together the connection of specific devices with actions such as selecting the operands and targets of an operation (who copies what to where?), specifying the relative spatial orientation of the devices, and choosing from multiple types of cross-device tasks to perform. Even a few seconds of silence during a social exchange quickly becomes awkward and uncomfortable, so it is essential that co-located users can perform the desired interactions quickly and with minimal distraction from the primary task of human-human communication.

Co-located users share physical space. Below, we argue that designs for spontaneous multi-device environments must carefully consider the sociological issues of shared space. Later in this paper, we present specific points we have explored in the design space of ad-hoc formation of multi-device environments. Our designs succeed in addressing some of these issues, but sometimes also fail with respect to others, and thereby teach us the design tradeoffs inherent to the approaches we have explored.

1.1. Proxemics - How People Share Physical Space

Proxemics is the study of how people use the invisible bubble of space that surrounds an individual (Hall, E. T., 1966; Sommer, R., 1969; Altman, I., 1975). For example, the next time you are at an airport terminal, sit next to a stranger when it is not necessary to do so. It will likely make you and the person next to you extremely uncomfortable. To amplify the tension, choose a person of the opposite gender, a different culture, or someone who is dressed very differently than you are. Time how long it is before the other person starts to show signs of tension or anxiety, and ultimately gets up and leaves. Sociologists have conducted many such "spatial invasion" studies (Sommer, R., 1969). In our example, your intrusion at the airport greatly increases the probability that the other person will relocate to a different area of the terminal within the next few minutes. The bubbles of space surrounding individuals may be imaginary, but social responses to violations of these spaces are real and deeply ingrained in people.

Thus, even when modern technologies and devices are involved, users still apply the social grammars of interpersonal distance to co-located collaboration. For example, social issues influence the role of technology in face-to-face consultations (Rodden, Rogers, Halloran, & Taylor, 2003). As another example, Scott proposes territory-based

techniques for tabletop interaction (Scott, 2003; Scott, Grant, & Mandryk, 2003). In our research, we have found that proxemic serves as an organizing principle for the different systems and approaches that we have explored.

Hall classifies proxemic distances between people (Hall, E. T., 1966), with *intimate* and *personal* distances within arm's reach, and *social* and *public* distances beyond arm's reach (Figure 1). Factors such as how well people know one another, or the presence of intervening obstacles such as a table or physical object, influence the selection of a social distance that feels comfortable to both people. Touching is particularly unwelcome in non-contact cultures (Altman, I., 1975). When close proximity or contact is necessary, people avoid continuous close contact and seek to synchronize their actions. For example, during a purchase from a cashier, each person's hand simultaneously moves towards the other to exchange monies, followed by relaxation of the increasing social tension by quickly exiting the personal space of the other[1]. Territorial behavior can extend to personal property and even to temporary possessions such as a parking space, or a chair in a waiting room (Deasy, C., & Lasswell, T., 1985). Personal space may extend to inanimate objects such as mobile devices; users may view them as an extension of their body while holding them.

Figure 1 ABOUT HERE

The lesson of proxemics is not to simply "avoid contact" or to discourage close proximity between users. No one distance between persons is "best" in all situations, and there may be situations where near or actual contact between users and devices is

[1] Personal communication, Geoffrey T. Raymond, Assistant Professor of Sociology, University of California Santa Barbara.

necessary or desired. Users who know each other well may want to work closely together, but strangers may want to exchange files while maintaining social distance. As Hall writes, "what is desirable is flexibility...so that there is a variety of spaces, and people can be involved or not, as the occasion and mood demand" (Hall, E. T., 1966, p. 110). While Hall wrote this in the 1960s, when personal mobile computers were certainly far from his thoughts, the lesson still applies when contemplating foundational principles for technology supported co-located collaboration.

Furthermore, communication patterns vary depending on relative body orientation (Sommer, R., 1969; Deasy, C. & Lasswell, T., 1985). Sommer shows that preference for face-to-face, shoulder-to-shoulder, or corner-to-corner seating arrangements depend on the task context. The collaboration literature emphasizes face-to-face interaction, but Sommer notes that corner seating preserves closeness while avoiding excessive eye contact, whereas students studying together strongly prefer side-to-side seating. When asked to choose a seating arrangement, the choice depends on the scenario (e.g., meeting with a rival, cooperation with a student, or casual conversation with an acquaintance). This suggests that system design for co-located collaboration should offer flexibility in terms of relative body orientation (and by proxy, device orientation), as well as functionality suited to different categories of interpersonal distance (intimate, personal, social, and public).

The proxemics issues summarized above underscore the evolution of our designs for addressing the problem of spontaneous connection of mobile devices. The role of an ad-hoc multi-display environment is not to assume or force a particular arrangement, but rather to support a variety of spatial and social relationships between users.

1.2. Synchronous Gestures

Our high-level approach to the ad-hoc multi-display environment problem has been to leverage *synchronous gestures* sensed across multiple distributed devices, possibly performed by multiple users, as a way to identify which of a potentially large set of candidate devices to connect for cross-device interaction. Synchronous gestures are patterns of activity that occur across a distributed system and assume a new meaning when they occur together in time. These patterns may occur in parallel, may be partially overlapped, or may even occur in a particular sequence.

Early examples of synchronous gestures in the literature include holding two devices together and shaking them (Holmquist et al., 2001), bumping a pair of devices together (Hinckley, 2003), or simultaneously pressing a *Sync* button on each device (Rekimoto, 2004). Different devices or participants contribute complementary portions of a signal, and the distributed system recognizes this composite signal only when it brings these portions together. Synchronous gestures require that the signal is unlikely to co-occur accidentally for nearby devices. The distance over which devices are considered "nearby" may be limited implicitly by short-range wireless radio communication, or through coarse-grained location sensing via wireless signal strengths (Bahl, P., & Padmanabhan, V., 2000; Krumm, J., & Hinckley, K., 2004). Synchronous gestures thus enable users to "name" the devices involved without knowing either device's IP address or network name. Synchronous gestures maintain some of the tangible feedback of establishing a wired or physical connection, while allowing suitable devices to find each other automatically. Yet users remain firmly in control of when and how to connect or disconnect the devices.

Synchronous gestures offer a compelling approach to the ad-hoc multi-display environment problem for several reasons. First, synchronous gestures do not require instrumenting the interaction environment or tagging each mobile device. Sensors and inputs that already exist on devices for single-user interaction can support collaborative interaction by identifying distinctive input patterns that occur across multiple devices. The participating devices act as an implicit sensor network; when a synchronous gesture pattern is recognized, it identifies the specific devices involved in an interaction without forcing the user to match symbolic names to the devices. However, this does require the devices involved to broadcast messages over a common network transport, or to exchange messages through an arbiter, in order to aggregate the necessary interaction events. One consequence of this sensor-agnostic nature is that the synchronous gesture approach is broadly applicable to different inputs and devices, as demonstrated by the projects we discuss in this paper.

Second, the user remains in control of his personal device at all times and can be certain that his device is not accessible to others unless he performs, or allows another person to perform, a synchronous gesture on his device. Since the device is not tagged, the user also knows that the device is not identifiable by a sensing system when the device is turned off.

Third, synchronous gestures offer the possibility to relax the constraints of intimate physical proximity and fixed relative orientation. Physical transport mechanisms such as RFID sensors, Infrared Data Association (IRDA) port connections, Near Field Communication (NFC), and other approaches that physically juxtapose devices (e.g. Ullmer, B., Ishii, H., & Glas, D., 1998; Tandler, P. et al., 1998; Rekimoto, J., et al., 2003)

require such constraints. Cooperative stitching, which we will discuss in detail later in this paper, is one example of a synchronous gesture for pen-operated devices that does not require close proximity of devices. One user draws a pen stroke and holds his pen at the top of the screen to "offer" a document, and a second user then pulls down from the top of his screen to "accept" the document. A different user on a different device performs each half of the gesture. The technique thus supports a variety of proxemic distances and relative orientations between devices and their users.

Finally, synchronous gestures leverage natural metaphors for synchronizing social activity, such as clinking glasses together for a toast, handing a document to another person, or placing a document on a meeting table to make it available to others. Furthermore, several of the synchronous gestures that we explore naturally offer rich information about how the devices are being connected. This is not the case with RFID sensing, for example, which detects that devices are in range of one another, but cannot determine which device "initiated" the encounter. This means that the user must perform additional steps to tell the computer whether information is to be sent or received across a given connection. In general, this observation led us to an approach that considers not just the connection step, but also the overall *connection-action phrase* that specifies how the connection fits into the larger workflow of the user's task. For example, when users join devices, they not only specify the devices to connect, but may also indicate the type (task-specific purpose) of the connection, the information that is to be shared, whether the information is to be sent or received, as well as the relative spatial orientation between the devices. Offering synchronous gestures in the context of connection-action phrases

enables the quick and facile creation of rich and flexible device associations that better suit the higher-level collaborative goals of the user.

2. RELATED WORK

Facilitating interactions distributed across multiple devices or multiple users, as well as techniques to combine two or more devices in an ad-hoc fashion, have been the subject of a number of related research efforts.

2.1. Related Research for Synchronous Gestures

Early examples of synchronous gestures (Hinckley, K., 2003b) in the literature include Smart-Its Friends (Holmquist et al., 2001) and SyncTap (Rekimoto, 2004). Smart-Its Friends associates a pair of devices when a user holds them together and shakes them. An accelerometer on each device senses the corresponding shaking patterns. When such "friend" devices enter the range of each other's low-power radio, they beep to notify users of the other's presence. A related approach displays a sequence of motions that a user must match by gesturing with an accelerometer-equipped device (Patel, S., Pierce, J., & Abowd, G., 2004). Researchers have also explored the possibility of using accelerometers to determine if the same person is carrying two devices (Lester, J., Hannaford, B. & Borriello, G., 2004).

SyncTap (Rekimoto, 2004) connects a pair of devices when one user presses a "sync" button on both devices at the same time (within about 30 milliseconds). If more than one sync operation occurs at the same time, SyncTap treats it as a "collision" and the user must repeat the operation. SyncTap's strength is that it offers a lowest common denominator solution, since nearly any device could include a suitable button. However,

a simultaneous button press provides little information about *how* to connect the devices, so the user must take additional steps to indicate the purpose of the connection. For example, to establish cursor migration between a laptop and a desktop display, a user performs a SyncTap gesture, and then switches to the mouse on the desktop computer to indicate which the edge of the screen to link to the mobile device. The BlueRendezvous project that we describe later in this paper demonstrates how current Bluetooth-enabled phones can build on the synchronous button press concept, by leveraging our approach of offering connection in the context of selection-action phrases.

Cooperative gestures are multi-person gestures for interaction on a single touch-sensitive tabletop device (Morris, Huang, Paepcke, & Winograd, 2006). A single-user gesture (such as deleting an object) may take on new meaning when performed by multiple users (e.g., delete all the contents on the table). More than one user may perform some of the synchronous gestures that we explore in this paper, but we emphasize cross-device collaboration rather than multi-user interaction with a device.

2.2. Physical Transports & Tagging

Users can employ physical objects to connect or transport information between devices. The mediaBlocks project (Ullmer, B., Ishii, H., & Glas, D., 1998) uses RFID-tagged physical blocks as tangible containers for data. The system uses a tag's ID to retrieve the "contained" information from the network cloud when the user docks the block to a different device. Physical objects with embedded RFID tags can also enable novel combinations of devices and functionality (Want, R., et al., 1999). In the tranSticks system (Ayatsuka & Rekimoto, 2005), a user can bind a pair of memory flash drives and then later plug them into two separate devices to create a connection. Although physical

transports are effective for some operations, physical constraints of real-world objects can limit the approach. A physical object cannot attach to more than one device at a time, nor can it move beyond arm's length without passing it to others, or walking across the room with it. The user must also switch their hands and attention back and forth between interacting with digital content on the screen, and handling the physical transport containers.

Cameras can sense visually coded tags on mobile devices and thereby identify the position and orientation of each device in an augmented meeting room (Rekimoto, J., Saitoh, M., 1999). A projector enables the environment to project graphical feedback showing an augmented working surface superimposed on the space occupied by the devices. Alternatively, this type of information enables pointing to identify desired devices (Swindells et al., 2002). Many mobile devices now contain cameras, so tagging the environment offers another approach. For example, a wall display can show a visual pattern that the user captures with their camera phone; decoding the pattern allows the phone to access a network service on the display (Scott et al., 2005).

2.3. Proximity and Near Field Communication

A system can provide some multi-device services by discovering nearby devices. Proximal selection (Schilit, R., Adams, N., & Want, R., 1994) allows a user to perform actions based on proximity, such as printing a document on a nearby printer. Wireless connection standards such as Bluetooth include mechanisms to discover other devices that are within range. The RADAR technique senses location via triangulation of wireless network signal strengths (Bahl, P., & Padmanabhan, V., 2000), but requires extensive calibration data. The NearMe system senses proximity of devices, rather than absolute

location, and thus removes the need for calibration (Krumm, J., & Hinckley, K., 2004). The Relate system (Hazas et al., 2005) allows for devices to discover their relative position and orientation to one another using a peer-to-peer infrastructure. In order to function, every participating device requires a Relate hardware dongle, which uses ultrasound technology to detect its location relative to other dongles.

Although these approaches enumerate nearby devices, they lack any direct means to identify a *specific* device; the user must perform additional operations to choose a device from a list of candidate devices. Even if a list uses names that people can understand**,** as the list grows longer, it becomes unclear which physical devices correspond to the names in the list.

Proximity sensing plays an important role for synchronous gestures because it limits the scope of devices that a distributed system must consider. For example, later in this paper we describe our *BlueRendezvous* application for smart-phones, which combines Bluetooth's automatic device discovery protocol with a synchronous button press. This enables connection to a specific proximal device. Our implementations of the Stitching (and Collaborative Stitching) techniques, also discussed later in this paper, employ 802.11 wireless networking, and use the NearMe system to limit synchronous partners to proximal devices (within about 20 meters).

Near field communication (NFC) techniques can support device association if each device is equipped with RF tags (Tandler, P. et al., 1998; Rekimoto, J., et al., 2003). Typically this entails the use of short range (0 to 3 inches) RFID readers that require placing a tagged device on a small reading surface. Long-range RFID readers present the

opposite problem: it is difficult for a user to judge whether a given tag lies within reading range. This uncertainty leads to breakdowns in interaction, as well as privacy and security concerns.

The Infrared Data Association (IrDA) standard avoids these problems by employing infrared sensor/transmitters to transfer data between devices. However, this can be cumbersome to use during a meeting because the users must place their devices close to one another and carefully align them in order for the infrared sensors to communicate.

2.4. Pick and Drop

Pick and Drop (Rekimoto, 1997) allows users to pick (copy) an item from one interactive display and drop (paste) it onto the display of another device nearby. Pick and Drop requires a pen embedded with a unique ID in order to detect when the pen from one device enters the range of another device. Identification codes are available on some pens, such as desktop display tablets from Wacom that sense 64-bit "pen-ID" codes, but pen-ID is presently not supported by any Tablet PC's or other mobile devices.

Because Pick and Drop relies on a unique ID bound to the pen, one user must employ the same pen for both the *Pick* and the *Drop* portions of the interaction. Short of physically handing the pen to another person, Pick and Drop cannot support a division of labor of the connection gesture across two separate users. This makes it difficult to form a one-to-many connection, or to form a beyond-arms-length connection. Furthermore, Pick and Drop does not coexist well with some natural uses of the pen: it becomes ambiguous as to whether the user intended to perform a cross-device operation, or if the user merely intended to point at or interact with the other screen.

Our research also moves beyond Pick and Drop by contributing the notion of a connection-action phrase. Pick and Drop provides a natural means to connect two devices that both offer a suitable pen-ID capability, but it does not integrate secondary information such as the relative spatial orientation of the devices, or a means to specify the purpose of the connection. Both of these additional pieces are necessary for a scenario where users wish to link their displays together to enable joint viewing of a large document or image that spans the displays, for example. The Pick and Drop work also does not consider the sociological implications of users collaborating in close quarters.

3. BUMPING AS A SYNCHRONOUS GESTURE

Our first foray into ad-hoc multi-display environments explored bumping together a pair of accelerometer-augmented tablet computers (Hinckley, 2003). We constructed a custom sensing module to provide a two-axis linear accelerometer as well as secondary supporting sensor data, such as touch and proximity sensors, that we used to help disambiguate which (if any) device was being held by a user. Accelerometers are now commonplace on devices such as Apple Computer's iPhone, the OQO Model 02, and the Toshiba Portege series Tablet PC's. The bumping interaction arose from our research into sensor-based interactions for mobile devices (Hinckley, K., Pierce, J., Sinclair, M., & Horvitz, E., 2000). Some of the interactions we envisioned appear on the iPhone device, for example, which uses an accelerometer to sense changes in the viewing orientation of the screen, as well as a proximity sensor to detect when the user is listening to the device. In short, sensor-enhanced mobile devices are likely here to stay. Interactions such as bumping devices together no longer seem as fanciful as they did just a few years ago.

In our system, bumping primarily supports *dynamic display tiling*, a technique that allows users to tile together the displays of multiple Tablet PC's by bumping a tablet into another one lying flat on a desk surface (Figure 2). This technique allows the "held" tablet to annex the screen real estate of the "stationary" tablet resting on the desk surface. For example, if the held tablet displays a large image when it bumps into a stationary tablet, the image expands to encompass both screens. Note that the system must be able to sense which tablets have been brought together, as well as which edges of those tablets are involved, in order for each tablet to display the correct portion of the image at the correct orientation.

Figure 2 ABOUT HERE

3.1. Detection of Bumping

With the two-axis accelerometer mounted in the plane of the device's screen, hitting one edge of a tablet excites one axis of the accelerometer much more than the other. This fact allows us to distinguish not only the edge struck, but also the direction of a bump, by observing the sensing axis with the largest response. Figure 3 shows a plot of the characteristic "equal and opposite" forces that are generated when a user bumps the left side of a held tablet ("local device") into the right side of another tablet ("remote device") resting on a table.

Figure 3 ABOUT HERE

Synchronous gestures require an arbiter that collates the distributed sensor data and determines when a complete synchronous gesture has occurred. The arbiter is an independent process that other applications must subscribe to if they wish to send or

receive synchronous gesture events. The implementation issues and challenges surrounding the arbiter are discussed in further detail in section 3.2.

Figure 4 illustrates the components and steps involved in detecting a bumping synchronous gesture (step 1). In our system, a synchronous gesture server at a well-known network address receives time-stamped events that mobile devices in the environment stream to it (2). The server synchronizes the time stamps and looks for matching sensor events such as complementary spikes in the accelerometer data (3). When the server finds a match, it sends a synchronous gesture event that informs the participating devices of each other's address (4) as well as the edge and direction of the bump so that the displays can be linked correctly (step 5).

Figure 4 ABOUT HERE

The system displays animated graphical feedback that shows an arrow sliding from the screen that initiated the connection onto the screen of the other device, with associated "snap-together" audio cues. Auditory feedback is important for multi-user interaction since the system cannot assume that both users are looking at their mobile devices when the interaction occurs. Even if one user is fully aware of the interaction, it is important to draw the attention of the other user to it as well.

Our system also uses the accelerometer to detect when the user pulls the devices apart. If a device senses substantial movement in any direction, it sends a notification to its partner device and disconnects. Alternatively, to allow users to hold or move the devices after connecting them, the disconnection can be delayed until the system senses the user walking away with a device, by using the accelerometer to sense a gait pattern

over several seconds (Hinckley, K., 2003a). Giving users facile means to control the state of a connection, with strong feedback as to the connectedness, are important design elements of synchronous gestures that we believe can ease privacy concerns when creating tightly bound links between co-located devices.

Bumping naturally phrases together several parameters of the cross-device operation, including the distinction of the sender from the receiver, and the desired geometrical arrangement of the devices, in a simple physical act that seems like a single cognitive chunk from the user's perspective (Buxton, 1986). For example, bumping is a more expressive connection-action phrase than just placing two devices in proximity of one another, or even connecting them with a network cable.

3.2. Arbiter's Role

There are a number of important aspects to consider that can affect the arbiter's reliability or performance, which in turn affects our system's capacity to detect and react to synchronous gestures events in a timely fashion. For example, the number of devices injecting sensor information into an arbiter is a factor that can degrade an arbiter's responsiveness since:

a) Comparing pairs of signals from a large group of devices can become computationally expensive;

b) The probability of collisions (i.e., when a signal corresponds more than one possible set of devices) increases as the number of simultaneous interactions among the co-located devices grows; and

c) The probability of reporting false positives increases as the number of simultaneous interactions among the co-located devices grows.

These issues are potentially problematic, but can be limited with careful design choices. The number of comparisons the arbiter needs to perform grows quadratically with the number of devices it monitors. For a set of a hundred devices that all are generating events at the same time, five thousand comparisons must be done— an operation that is not prohibitive for modern devices. Using a proximal arbiter that only considers devices within a certain distance is a straightforward way of limiting the number of devices that must be considered simultaneously. Ideally, if device hardware supports modulation of wireless radio signal transmission strength, a device could attenuate its signal so as to limit communication to a computationally feasible set of devices.

The arbiter knows when collisions (potentially ambiguous synchronous gestures) occur. The arbiter may not be able to determine what the desired connections are, but it does know which devices are involved in the ambiguity. SyncTap addresses collisions (e.g, two users performing simultaneous button presses at exactly the same instant) by simply discarding the ambiguous events, forcing the users to rearticulate the gesture. But the arbiter can inform the participating devices of the collision so that users may be informed of the failed attempt, and prompted to try again. If collisions are infrequent, this is a reasonable solution.

When communications are limited to proximal devices, another possibility is to fall back on social protocol. For example, users may coordinate their activities and take turns when potentially ambiguous actions might occur.

The use of a proximal synchronous gesture arbiter does raise some difficult systems issues that would have to be addressed for synchronous gestures to be broadly deployed. For example, a centralized arbiter might not be available at a particular location. Although we did not attempt to implement a decentralized system, we can envision an architecture where a set of devices essentially acts as a decentralized sensor net, where any node (or more than one node) can assume the role of an arbiter. Developments in sensor nets and mesh networking (Mir, 2006) point suggest that this should be possible, but we have not explored the feasibility of such approaches for sensing synchronous gestures.

Another issue is that a user might not trust the arbiter's credentials (e.g., while at a meeting in a rival company's conference room). We remain mindful of trustworthiness issues and believe there are several ways to address them. Rekimoto describes success using simple cryptographic protocols to avoid "man in the middle" attacks (Rekimoto, 2004). Also, for synchronous gestures, social protocol is a crucial part of the connection exchange, and constitutes an information channel that is hard to eavesdrop by entities outside the proxemic circle in which the interaction occurs. In our work, the arbiter facilitates the creation of a tunnel between devices, i.e. once the connection occurs, all data transmission occurs directly between the participating devices without passing through the arbiter (Figure 4.5). Thus, the arbiter learns only of the connection between the devices, but is not privy to the information passed between devices. If the arbiter

reports a malicious address for a "partner" device, then the users will be able observe that the desired devices are not connected to one another. Nonetheless, if it is not acceptable to rely on social protocol as a failsafe, then synchronous gestures may not be an appropriate technique for an untrusted environment.

Our implementation uses a fixed arbiter at a well-known address. The arbiter determines which devices to synchronize based on proximity as sensed by the NearMe system (Krumm, J., & Hinckley, K., 2004). We employ a client/server architecture with a wireless network for data transport, but our emphasis is on exploring the interaction paradigm rather than implementing an elegant distributed sensing architecture. For example, our system cannot handle issues such as the sudden change of a device's network addresses due to transient wireless dropouts or handoff, or migration of the "arbiter" to different devices or servers as various mobile devices come and go. These issues, and the challenges outlined above, were not resolved by the arbiter that we used to prototype the techniques described in this paper. These remain important systems problems that future work will need to resolve if synchronous gestures are to become commonplace.

3.3. Social Implications of Bumping

Bumping requires that the users' devices are proximate enough to contact each other, but the users can avoid touching one another because each device acts a physical barrier between the users, and only the devices must touch. Bumping also leverages the socially acceptable custom of clinking glasses together, for example. Our dynamic display tiling technique supports a proxemic *intimate* connection between devices, where they are left in close proximity so as to provide a joint display that spans the screens of the two mobile

devices. While users may desire this level of closeness for some tasks, on other occasions it is not appropriate. However, even though the bumping gesture itself requires contact between devices, the system can discern some variations of the gesture. For example, if two users jointly bump their devices together while each is held in mid-air (as opposed to while one device is resting on a flat surface), the system offers an alternative transaction that by default exchanges the users' home web pages and shows each on the screen of the other device. In another variation, one user can tilt their tablet at an angle while bringing them together, to "pour" information on the system clipboard from one device to the other. For these transactions, the contact is transient and facilitates a proxemic *personal*, rather than *intimate*, connection between the devices.

4. ARMS-LENGTH STITCHING

In this section, we explore the properties of drawing a pen gesture that spans multiple displays as a synchronous gesture to link multiple pen-operated devices (as well as touch-sensitive devices that can sense contact from both pen and finger inputs). While employing a pen to link devices is arguably less direct and less tangible than bumping devices together, it offers an interesting approach because the pen can perform other operations on the device as well. With bumping, it is necessary to physically handle and move the mobile device, so the user's hands are occupied by the device itself. With a pen, however, the user can fluidly integrate both within-device and cross-device operations. This enables rich connection-action phrases that would be difficult to support directly with the bumping technique.

4.1. Connection-Action Phrasing with Arms-Length Stitching

We originally called this technique "stitching" (Hinckley et al., 2004), but here, we will refer to it as arms-length stitching to distinguish it from the cooperative stitching technique that we will discuss in a subsequent section of this paper. To perform an arms-length stitch, a user starts moving the pen on one screen, crosses over the bezel, and finishes the stroke on the screen of a nearby device (Figure 5, Figure 6). The arbiter component of the system observes the properties of each portion of the pen stroke, synchronizes them via wireless network communication, and recognizes the stitching gesture as a unitary act performed by one user, thus binding together the devices.

Figure 5 ABOUT HERE

We explore arms-length stitching in the context of a photo-sharing application known as *StitchMaster*. This application enables users to drag photos from one device to another nearby device in a manner analogous to drag and drop on a single screen. The application also offers other connection operations, such as displaying a photo across multiple screens, projecting the photos that one user selects in a full-screen view on the other user's tablet, or establishing a persistent shared workspace across the devices.

Arms-length stitching in *StitchMaster* integrates the operands of a cross-device operation by beginning the gesture with a lasso around the desired objects. The user can pause briefly, or make a sharp turn with the pen, and then continue the lasso until the pen hits the bezel of the screen. The initiator component of the system segments the gesture into a *selection lasso* and a *first-half stitch* at the resulting inflection point.

To continue an arms-length stitch, the user lifts the pen, places it back down at the closest edge of the desired device, and draws a *second-half stitch* in an approximately straight-line continuation of the first-half stitch. The user can then lift the pen, which indicates that a default cross-device operation with the selection should occur (in *StitchMaster*, this copies the selected photos to the other device).

Figure 6 ABOUT HERE

If the user holds the pen still at the end of the second-half stitch, this brings up a marking menu (Kurtenbach & Buxton, 1993) with different options for how to connect the device. Figure 7 summarizes the main cross-device commands supported by *StitchMaster*. Since the user invokes the marking menu at the end of the operation, on the screen of the other device, we refer to this as a *postfix remote menu*. We chose to implement cross-device menus in this manner because the arbiter component of the system does not know what operations a pair of devices can support until it knows which devices the user is connecting.

Figure 7 ABOUT HERE

However, as noted in our overview of proxemics, and as we observed in usability studies of arms-length stitching, users are hesitant to remain in the personal space of another user's device any longer than is necessary. For example, we observed that a user receiving photos would lean away from his own screen as the other person reached onto it to draw the second-half stitch. The user sending the photos tended to draw very short second-half stitches and then quickly lift the pen. For these reasons, accessing alternative

connection commands by requiring the user to pause and then interact with a menu on the remote device was not, in retrospect, the best design decision.

We now believe a better design would be to dispense with the notion of a "default" operation in lifting the pen, and present a *postfix local menu* on the sender's screen to choose the desired operation after completing the arms-length stitch itself. This would enable the sender to establish the connection, but then quickly exit the space of the other device and relax the social tension caused by the temporary invasion of space. Another alternative would be to integrate *prefix local menus* with the first-half stitch, e.g. by pausing at the edge of the screen. However, since the identity of the second device is still unknown at this point, a prefix local menu would not be able to present commands that depended on the type of devices being connected. Thus, we believe postfix local menus offer the best design compromise.

StitchMaster allows users to create a *persistent shared workspace*. In our original implementation, this was achieved by drawing an arms-length stitch between screens without lassoing any photos to share, and without waiting for the menu to appear. *StitchMaster* draws a red frame that appears to span the two screens to give users ongoing feedback that the two devices are in fact connected. While in this state, *StichMaster* supports a technique known as *the transporter*: a user can drag photos to the edge of his screen, and then pause. The initiator component of the system displays a short animation of a collapsing blue square, after which it plays a sound effect and transports the photos to other device. The user can back out of the operation before the animation finishes by continuing to drag (rather than continuing to pause). The persistent shared workspace with the transporter mechanism remains active until the devices move beyond wireless

range of each other. Either user has veto power over the connection and can close the workspace by choosing *Disconnect* from a menu.

4.2. Recognition of Arms-Length Stitching Gestures

Our implementation of stitching uses a synchronous gesture server similar to that used to detect bumping gestures (Hinckley, 2003). The stitching server recognizes a stitch by looking at the patterns of pen activity from each pair of participating devices. We define an *envelope* as the time interval during which the pen is in range of the screen and is moving at a speed above a predetermined threshold. The stitching server then looks for two consecutive envelopes from a pair of devices that match a specific pattern, as illustrated in Figure 6. The specific criteria used to recognize this pattern are:

a) The first envelope must end near the first screen's border and last longer than a timeout (250 ms). Likewise, the second envelope must start near the second screen's border, and last longer than another timeout (100ms).

b) The second envelope must start after the first envelope and no longer than 1.5s after the first envelope. This time interval is long enough to support stitching between tablets within arm's reach.

c) If multiple envelopes occur at the same time, our arms-length stitching implementation considers the connection to be ambiguous and the user must repeat the gesture.

For step (c) above, we also have experimented with restricting the match between line segments based on the angle of incidence with the screen edges, but one drawback of this

approach is that some valid stitching gestures can be rejected due to incidental deviations in the angle at which the user draws each half of the stitching gesture. In Section 5, where we discuss cooperative stitching, we address this issue by providing feedback and quick gestures to filter potential recipients before forming the action connection.

We found these criteria suffice to recognize intentionally executed stitching gestures, but just as importantly, they help to minimize false positives. Incidental pen motions from two users concurrently using pens rarely satisfy these criteria.

In our original implementation of arms-length stitching, we allowed stitching gestures to be drawn either in-air (i.e. in the hover-sensing state of the Tablet PC), or via dragging the pen in contact with the screen. We eventually settled on using pen contact for all stitching gestures, as it is less likely to occur by accident, it offers a more self-explanatory interaction for users, and it provides a consistent gesture for devices that do not sense a hover state (e.g. handheld devices with resistive touchscreens). We now disable in-air stitching by default, a choice that further decreases the possibility of accidental stitching gestuers.

For example, let us assume a simple model where a stitch occurs when the arbiter detects a stitch when it sees a *lift_pen* at the edge of device A, a *land_pen* at the edge of device B within a small time-window. Then the chance of an accidental stitch is roughly:

P(accidental_stitch) = P(lift_pen).P(pen_at_edge).P(land_pen).P(pen_at_edge)

Because a stroke contains a *lift_pen* and a *land_pen*, *P(lift_pen) = P(land_pen)*. Likewise *P(pen_at_edge)* should be the same regardless of the device. We analyze pen

data (Grossman, Hinckley, Baudisch, Agrawala, & Balakrishnan, 2006) reflecting people's usage of Microsoft's Journal application and we find that *P(lift_pen)* ≈ 0.12 and *P(pen_at_edge)* ≈ 0.05. These numbers yield a chance of an accidental stitch of roughly 0.0036%. This number is likely to be even lower if we consider the stricter conditions under which a stitching should occur, e.g., minimum stroke length, stroke curvature, etc.

4.3. Determining the Geometry of Arms-Length Stitching

An important contribution of arms-length stitching is that it uses the absolute geometrical information of the pen's location on each screen to compute a transformation describing the approximate spatial relationship between the connecting devices (Figure 8). This transformation enables the connected displays to present an automatically calibrated unified surface across them, so that graphical feedback appears to span such surface. StitchMaster employs this feedback to indicate as clearly as possible, exactly where photos are coming from or where they are being copied to (as seen in Figure 5 and Figure 6)

Figure 8 ABOUT HERE

Furthermore, our automatic calibration technique allows the participating devices to present a convincing illusion that StitchMaster's workspace spans their displays. For example, once devices are connected, the user can drag an image so that it appears partially on one screen, and partially on the other screen; the other user can then "grab" the portion of the image on his screen to take it. Note that this offers one way for users to avoid reaching onto one another's screens once the connection itself has been formed. Our original system only supported arms-length stitching between devices at the same

screen orientation, but we subsequently extended the technique to include rotational transformations for shoulder-to-shoulder, face-to-face, and corner-to-corner arrangements of devices (Figure 8). Thus, our technique for automatically computing the geometry of the displays is an essential element to provide users with flexibility in the relative body orientation (and device orientation) with which they engage other users.

Our geometrical transformation technique determines the relative orientation of the devices, but it lacks sufficient information to accurately determine how far apart the devices are (we did experiment with using the time delta between the two halves of the stitching gesture as a proxy for distance, but found the results to be unsatisfactory). Despite this limitation, the illusion of cross-device feedback and of cross-device application content remains quite convincing to users; nobody in our user tests noticed this quirk, even though many of the users employed the devices while separated by a foot or more.

The geometric calibration uses straightforward trigonometry based on the absolute position of the pen trace on each screen to derive a coordinate transformation that maps points in a remote device's screen space to the coordinate system of the local device's screen. Since our original research, we have extended this transformation to support all 16 possible (four different orientations for each of a device's four edges) 90-degree rotations between devices. Readers uninterested in how to compute this transformation can browse the figures and safely skip to the next section.

We deconstruct the general calibration problem into two simpler ones. First, we determine the coordinate transformation in a canonical orientation (in which both devices

are “facing north” and where the stitching gesture flows from left to right) while assuming that there is no rotation, as shown in Figure 8. Second, we apply any necessary rotation transformations to the canonical transformation computed in step 1 (Figure 9 illustrates an example of this 2nd step).

Figure 9 ABOUT HERE

Step 1 assumes the participating devices are in the canonical orientation. We then construct a coordinate transformation function $F_{i,j}(x, y)$ that transforms coordinates from device *i* to device *j*. $F_{i,j}(x, y)$ includes scaling factors to translate the pixel coordinates of a device into real-world units of distance (e.g., millimeters), thus allow stitching between devices with dissimilar dots-per-inch of pixel density.

With the transformation function $F_{i,j}$ in hand, all we need to solve the general calibration problem is to generate functions $T_i(x, y)$ and $T_j(x, y)$ that rotates the devices’ coordinates system into the canonical orientation “devices facing north, stitching left to right” case (Figure 9 illustrates how such pair of transforms affects an arbitrary device orientation). $T_i(x, y)$ is always one of four potential rotation transformations $R_k(x, y)$, that can be composed with $F_{i,j}(x, y)$ to compute the actual mapping $M_{i,j}(x, y)$ between the device *i* and device *j* ’s coordinate systems.

This refinement of our original geometrical calculation enabled us to implement stitching between any combination Tablet PC’s, Pocket PC’s (Figure 10), and a large-format touch-sensitive SmartBoard display in our labs. All of these devices can sense contact from the pen of a Tablet PC.

Figure 10 ABOUT HERE

4.4. Arms-Length Stitching Usability Study

We conducted a usability study of the StitchMaster photo-sharing application to assess if users could effectively use stitching gestures to perform multi-device operations. At the same time, we wanted to observe users' behavior and identify usability issues and concerns with arms-length stitching. We present some details of this study here as it motivates our subsequent work with cooperative stitching.

Pairs of users, seated side-by-side at a rectangle, collaborated in a free-form photo-sharing and sorting task. Twelve participants, six of whom had previously used pen-operated devices, were drawn from the general public. None of the paired participants knew each other prior to the study. We provided each participant a Toshiba Portegé 3500 Tablet PC running StitchMaster. Each session consisted of a brief practice session, where the participants learned basic pen operations such as selecting images and dragging them on the screen. Then the experimenter verbally explained how to use various features of the application, and asked participants to try them out one-by-one.

The experimenter first asked participants to "connect the devices by making a pen stroke across the devices" but did not show participants how to do this. With this instruction, all 12 participants, on their first or second try, successfully connected the devices. Next, the experimenter explained how to move photos between devices.

We did observe a few cases where arms-length stitching did not always operate as users expected. We also observed a few false positives triggered by the in-air style of stitching. These generally occurred when users failed to successfully stitch, and then

returned the pen to their screen (while accidentally moving in the in-air state) to try again. Thus, in subsequent work we abandoned the in-air approach.

Participants sometimes started a stroke too close to the edge of the screen; the short stroke left the arbiter component of the system with insufficient information to determine the desired connection. To address this, we added a visible margin of 2 cm to serve as user feedback of the minimum distance from the edge of a screen that a stitch must begin.

Participants were enthusiastic about the concept of arms-length stitching as embodied by StitchMaster, and overall participants strongly agreed that they would want to use such functionality if it were available to them. However, many participants expressed caveats about their control over security and privacy. For example, participants asked if "Once connected, can a person take my other stuff?" or if there was a "lock-out for security and privacy."

4.5. Social Implications of Arms-Length Stitching

During our study, several participants asked almost immediately if it was necessary to have the two devices right next to each other. The experimenter replied that it was not necessary, after which these users moved their devices apart to roughly 15-40cm of separation. Users were uncomfortable with leaving their device in prolonged contact with that of another user. This observation is consistent with sociological observations that in non-contact cultures, temporary access into one's personal space is sometimes permitted, but touching is taboo (Hall, 1966).

Clearly, arms-length stitching must support connections without requiring close contact. Arms-length stitching employs a time-out of 1.5 seconds between the first-half

stitch and the second-half stitch; this allows sufficient time to perform a stitching gesture between devices within arm's reach, up to a maximum of about 75 cm. However, it is still valuable for users to place their devices together. Users commented that they "liked the ability to split the view, so there are no two faces trying to peek at only one screen," and that "the wide screen [side-by-side connection] would be nice for collaboration, like for two people working on the same spreadsheet."

We also observed that participants engaged in an "establish and relax" pattern of activity where they were willing to temporarily invade the space of the other person's screen to establish a connection, but then they would exit the other person's space as soon as possible. This partly explains the popularity of the "transporter" interaction: when a cross-device gesture forms a persistent connection, it enables users to amortize the social cost of forming the initial connection over a longer session of sharing content back and forth. This may be one reason why StitchMaster's transporter interaction was a popular way to transfer additional photos once the users had established a persistent shared workspace connection via an initial arms-length stitching gesture.

Taken together, the techniques implemented by StitchMaster complement one another to provide the "flexibility [where] people can be involved or not," as advocated by Hall (Hall, E. T., 1966), in the ways that users digitally collaborated in a shared physical space:

- Intimate spaces: Placing two devices close together and then expanding a photo to fill both screens, for example, supports tight collaboration between friends or co-workers who may need to work together on a large document.

- Personal spaces: For a transient operation such as copying a photo to another device, users should not be required to keep their devices in close contact. As noted above, our implementation explicitly considers stitching to another device that is not immediately adjacent, but that still resides within arm's length.

- Social and public spaces: The main limitation of arm's length stitching is that it cannot support the creation of connections beyond arm's length. But it does enable interactions to transition from personal to social distance by supporting persistent connections with the transporter interaction metaphor to bridge the intervening physical distance. We observed that users verbally coordinate their actions when using the transporter; for example, one user announced "here's a care package for you" as he started to drag some photos into the transporter. This enables social coordination – the other person can respond "OK" to signal willingness to accept the transport, or "wait a second, I'm trying to finish this" to refuse it. Thus, the transporter works well for co-located users, but might not be suitable once users exit a shared physical space.

- Orientation of spaces. Our enhanced automatic calibration technique for determining the relative spatial orientation of devices permits consistent interactions that show feedback at the correct screen orientation on each user's device for all side-to-side, face-to-face, and corner-to-corner seating arrangements. We have also experimented with associating specific functions with specific relative orientations of the devices, such as providing a shared whiteboard application when users dock their devices at the face-to-face orientation (Hinckley, 2003a).

Thus, arms-length stitching provides a flexible interaction mechanism for connecting mobile devices in a variety of useful ways. In the following section, we introduce cooperative stitching, a new variation of arms-length stitching that permits the formation of connections beyond arms-length, as well as the formation of one-to-many connections between three or more users.

5. COOPERATIVE STITCHING

Cooperative stitching mimics the social protocol of handing a document to another person, but takes advantage of the virtual nature of the transaction to make it more flexible than passing around a physical object. In the real world, one person offers a document by extending it in hand, and a partner accepts it by reaching out to grasp the opposite side of the document. The document serves as a physical barrier between the users that prevents person-to-person contact, and holding out the physical document grounds the shared understanding that the document is being offered, and that it is permissible to take it.

Cooperative stitching distributes the two halves of a stitching gesture among a group of cooperating users (Figure 11), rather than requiring one user to draw both halves of the gesture. Cooperative stitching enables one user to offer a document by dragging it to the top of his screen, and holding it there with the pen. Another cooperating user can then drag down from the top of his screen to accept the document. The document is only available while the user offering it continues to hold his pen near the top of the screen. In contrast with the invasion of personal or even intimate space required by arms-length stitching, cooperative stitching can support connections between two or more devices beyond arm's length. Cooperative stitching is an example of a cooperative gesture

(Morris et al., 2006) that is distributed across multiple distributed devices, as well as multiple users.

Figure 11 ABOUT HERE

Cooperative stitching builds on the synchronous gestures we have described so far in a number of dimensions:

- It allows different users to act in concert in order to create both transient and permanent connections across multiple heterogeneous devices;
- It supports one-to-many connections among devices, as well as one-to-one transactions that may be desired between co-located users beyond arm's reach;
- It provides a lightweight mechanism to characterize the attributes of a connection, such as the type and hierarchy of the connection – i.e., who initiates the connection and who receives it;
- It takes advantage of social cues as a way to provide users with familiar engagement rules and to decrease the complexity of the interaction design by relying primarily on social arbitration rather than technical solutions to all issues;
- It addresses some of users' privacy concerns by providing them with feedback to see who accepts a connection and quick gestures to control who is (or is not) allowed to participate in a connection.

When one user initiates a stitch, cooperative stitching allows multiple users to complete the gesture to form a one-to-many connection. This is not possible with arms-length

stitching. Both the sender and all receivers see graphical feedback showing who the participants in the connection are, as well as a preview of any offered files. Either the sender or receiver may rescind the connection if desired (Figure 12).

Figure 12 ABOUT HERE

5.1 Design Space of Cooperative Stitching Gestures

Since cooperative stitching is intended for small co-located groups, users can employ social protocols such as body language, eye contact, and explicit verbal cues to coordinate their actions. This influences the relative timing the whole system can require for each user's contribution to a cooperative stitching gesture, as well as the degree of graphical feedback that may be necessary during a connection process. We considered three different approaches to the relative timings of the actions required to perform a cooperative stitching gesture:

Serial action: Arms-length stitching recognizes a pattern where a pen stroke starts on one screen, leaves the screen, and then finishes on a second screen. Two cooperating users could emulate this pattern by having each user draw half of the gesture in a strictly serial order. Arms-length stitching required both halves of the gesture to be performed within 1.5 seconds of one another, but this short delay might be insufficient to coordinate the actions of multiple users. However, using a longer time-out delays the completion of a connection (since the arbiter component of the system must wait until the full time-out expires before determining if all participants have joined the connection). Consequently choosing a time-out that satisfies these conflicting requirements is difficult.

Synchronous action: We considered following the example of SyncTap (Rekimoto, 2004) by requiring simultaneous synchronous stitching gestures from each user. Unlike SyncTap, stitching gestures have a direction (the sender pushes up, and the receiver pulls down) so there is no ambiguity as to who is sending and who is receiving. However, it may be difficult or unnatural for multiple users to precisely coordinate their pen gestures, particularly when the natural social grammar of gift-giving is for the giver to first offer the gift, and then for the taker to accept it.

Overlapped action: Cooperative stitching can allow for partially or fully overlapped actions on the part of the cooperating users. This avoids the requirement for precise synchronization, and also can avoid the need for an arbitrary time-out by allowing the sender to start a stitching gesture and then "hold" open the offer to connect (by keeping the pen on the screen) for as long as is necessary or desired.

We implement two variations of cooperative stitching to explore the design possibilities noted above, with the exception of the synchronous action approach, which we decided not to pursue further. The first, dubbed *Stitch+Lift* (Figure 11, top-left), is a serial action design. The sender draws a straight line (stitching gesture) to the top of the screen and lifts the pen. Any other user that wishes to participate then has 4 seconds to complete the stitch by drawing a straight line down from the top of his screen.

The second, dubbed *Stitch+Hold* (Figure 11, top-right), mimics the social protocol of handing an object to a person, and thus follows the overlapped action design approach. The sender draws a stitching gesture to the top of the screen and holds the pen at the edge

of the screen to offer a connection. Any other users can complete the stitch as long as the sender continues to hold the pen down.

Note that either of these designs can coexist with arms-length stitching, allowing the techniques to complement one another. The automatic calibration of the spatial arrangement of the screens that occurs during arms-length stitching is particularly useful for intimate/personal space interactions where it may be necessary to show feedback or images that span two displays placed close together. However, a one-to-many cooperative stitching gesture lacks this information (all users draw lines from the top of the screen, so there are no longer sufficient cues to determine the relative orientation between the sender and receiver), making the technique better suited to social/public space interactions. Furthermore, in early pilot testing we found that there was less need to show feedback that spans displays when displays are widely separated. We also observed that users sitting around a table naturally expect that "the group shared space is at the top [of the screen]," i.e. towards the center of the table. This is why cooperative stitching uses the top edge of the display as the place to offer or accept connections between devices.

5.2 Graphical Feedback for Cooperative Stitching Gestures

To make the state of the connection as clear as possible, and to address privacy and security issues raised by user comments in our arms-length stitching study, we augment cooperative stitching with graphical feedback throughout the different stages of the multi-user gesture. Our early pilot testing made it clear that a connecting solution must consider the perspective of different participants:

The sender (the user that initiates the stitch), who wishes to be made aware of *who* is accepting his offer of a connection; and

One or more recipients, each of whom wish to see *who* initiated the connection with *what* files or information.

We provide awareness of *who* participates in a stitching gesture by showing a tab containing an icon identifying the connection partner (Figure 13). The sender sees icons corresponding to the devices accepting the connecting. Each receiver sees an icon corresponding to *who* proposes the connection as well as what files the sender offers. These icons can be provided by the user's system, e.g. in the same way a buddy icon can propagate through different internet services once a user logs into a instant messaging system. Operating systems such as Windows XP and OSX associate such icons with a user's account.

Figure 13 ABOUT HERE

Exchanging identification icons in this way does not preclude adversarial users from falsely advertising their identity, e.g. by pretending to be someone they are not. While we did not focus on adversarial scenarios, straightforward extensions of our techniques could help to address such issues in future work. For example, the true sender might use the pen to mark a feature of his or her icon and use social protocol to inform trusted users of how he had marked his icon. In this scenario, trusted users can only complete the stitching gesture by grabbing the right icon from the right place. A similar strategy can be used for scenarios where the sender wants to make sure only trusted receivers connect, i.e., receivers specify icon features through social protocol that must be marked in order for a

connection to be completed. There is a tension between security, privacy and fluidity of use, with fluidity of use degrading as security mechanisms increase. The interaction techniques presented here are no exception.

Because of the distributed nature of the cooperative stitching interaction, we consider several possible designs for presentation of feedback:

Early feedback: When the sender initiates a cooperative stitch, the arbiter component of the system does not yet have any way to know which cooperating users will accept the stitch, yet it can broadcast the offer of the connection to other nearby devices. In the *early feedback* design, this broadcast causes a tab to drop down from the top of other co-located users' devices showing who is offering the connection. This reduces the need for users to rely on social protocols to form a connection, but the drawback of this approach is that users may broadcast offers to unintended recipients who in turn are distracted by undesired offers from nearby devices.

Intermediate feedback: The connection process can rely on the users to initiate cooperative stitching through social protocols, and defer feedback until recipients start drawing a stitch. In the *intermediate feedback* design, as soon as the system observes the first 100ms of the second half of a cooperative stitch, the system knows which user is accepting the connection. The receiver then reveals a drop-down tab showing who initiated the offer. As the accepting user continues to drag the pen, her system also shows a thumbnail preview (attached to the pen) of the incoming files. At this point the accepting user can either complete the stitch, or decide to refuse it, by dragging the files back to the top of the screen, or by crossing out the drop-down tab. Likewise, the sender

sees a drop-down tab as soon as a recipient starts accepting a cooperative gesture, and can cross out any undesired recipients (Figure 12). This provides each user with feedback as soon as it is possible to know what is being transferred to whom, and provides interaction mechanisms for either user to back out of the transaction if the feedback reveals a connection that is not actually desired.

Late feedback: The sender and receiver devices can wait until all users finish their cooperative stitching gestures to reveal feedback of who participates in a connection, and what files are transferred. However, we found during early pilot testing that users have a strong desire to see what they are accepting, and from whom, as soon as possible. Even though social protocol seems sufficient for users to coordinate their actions, users still want immediate reassurance that they are connecting only with the intended parties. Thus, we decided not to further pursue a *late feedback* design after the pilot testing stage.

5.3. Cooperative Stitching Usability Testing

We implemented versions of stitching, Stitch+Lift and Stitch+Hold techniques for a variety of platforms such as Tablet PCs, Pocket PCs and Smart™ boards, all of which could participate on a stitch connection – e.g., tablet to PDA, PDA to electronic whiteboard, etc. However, we gathered most of all heuristic evaluation and preliminary user study data through our implementations running on Toshiba Portegé 3500 Tablet PCs.

We conducted usability testing on the Stitch+Lift and Stitch+Hold cooperative stitching techniques with groups of 4 users, each equipped with a Toshiba Portegé 3500 Tablet PC, and seated around a 1.2 x 0.76 m rectangular table (Figure 14). The

Stitch+Hold technique employed the early feedback design strategy, while the Stitch+*Lift* technique employed the intermediate feedback strategy. Unfortunately, we did not test the Stich+Hold technique with intermediate feedback. In retrospect, this was a blunder as our results outlined below suggest this may have been the design users would have preferred.

Figure 14 ABOUT HERE

We also included two alternatives for users to compare to: (1) arms-length stitching, and (2) *list selection*, that is, selecting the names of the desired users from a list of proximal devices. The experimenter briefly demonstrated each technique, after which users performed a number of trials where they had to transfer icons to one, two, or all three of the other users seated at the table.

For the list selection condition, we systematically varied the list so that it presented 3 names (the names of the seated paticipants, with no "distractor" names), 19 names (3 users + 16 distractors), or 131 names (3 users + 128 distractors). This let us probe the scalability of the technique as the number of candidate proximal devices increases. We assigned each participant a fictitious name that they kept throughout the study, and the lists used these fictitious names plus other distractor names, rather than cryptic computer names. The list with only three names takes the optimistic view that proximity sensing technology will one day be able to identify only exactly which users are participating in a collocated group. The lists of 19 and 131 names represent scenarios where there are increasing numbers of nearby devices that might be candidates for connections. The 19-item list was small enough to view on a single screen without scrolling.

Overall, users strongly preferred the cooperative stitching techniques to arms-length stitching technique in this task context. For the arms-length stitching technique, the longest edge of the table was too far to reach the device at the opposite end, but we did observe that users would step closer or slide their device toward the other user to reduce the distance. User preferences split between the Stitch+Lift and Stitch+Hold cooperative stitching techniques. However, we observed that users would consistently gravitate to the Stitch+Hold manner of performing the cooperative stitching gesture once they had been exposed to it. Most users who preferred Stitch+Lift to Stitch+Hold found the broadcasted early feedback offered by Stitch+Hold to be distracting, and senders also felt uneasy that this might advertise the availability of a connection to undesired participants. Thus, even though our pilot users had felt that early feedback might be helpful, our usability testing of the final implementation did not support this. These observations suggest that Stitch+Hold with intermediate feedback offers the most suitable combination of design options for cooperative stitching.

If the connecting system knows exactly what persons are involved in a group, selecting participants from a list offers a very effective solution. The list with three names was the overall preferred technique for 5/7 participants (one user's ranking responses were lost). However, only 2/7 participants preferred the 19-item list to cooperative stitching, and most users found the very long 131 item list very cumbersome. Thus, lists are effective for a small number of candidate devices, but the solution scales very poorly as the number of proximal devices increases.

Participants made extensive use of social protocol in the form of verbal and physical cues to facilitate cooperative stitching. Participants were mindful of privacy and security

issues and responded favorably to the possibility of exerting veto power over a connection and its participants, but gestures to perform these operations were not yet functional at the time of the study. Based on this positive feedback, we implemented fine-grained connection controls for both sending and receiving devices.

5.4. Connection Control for Security and Privacy

In an ad-hoc connection between devices, people want to be sure that there are no eavesdroppers and only trusted or intended people participate. Conversely, people do not want to be flooded by unsolicited connection requests in a collocated environment. This generates a tension with our desire to offer simple and fluid connection techniques because adding complicated security or failsafe protocols can disrupt the interaction.

A mechanism that makes people feel aware and in control of the connection provides a good compromise between simplicity, security and privacy. Cooperative stitching provides the following controls over connections:

- The connection tabs that appear at the top of the screen show the sender *who* is accepting a connection, and allow the sender to select who *should* or *should not* be permitted to connect. By default, the system permits the connection unless the sender uses the pen to cross (Accot & Zhai, 2002; Apitz & Guimbretière, 2004) the undesired tabs (Figure 12).

- On the receiver's device, the connection tabs show *who* initiates the connection and *what* is being received. The receiver can also cross out tabs to reject a connection. The receive also has the option of dragging the pen back to the top of the screen to

rescind a stitching gesture that they have started, if the resultant feedback of the documents being received indicates that they are not desired.

This "veto power" has different consequences for the sending and receiving devices. While a veto on the sender severs a connection to a distrusted or unwanted device, on a receiver a veto could reject either a sender or just the information being sent. Such a distinction between levels of rejection can be important depending on the type of connection established. Figure 12 illustrates the case where a sender rejects an non-trusted receiver.

6. BLUERENDEZVOUS

This section discusses synchronous gestures in a SmartPhone application known as BlueRendezvous. Mobile phones are quickly becoming the predominant computing platform in many regions of the world. Consequently, a large community of people could benefit from direct ways to connect their phones to exchange information, collaborate, or leverage the functionalities of each other's devices. We have deployed BlueRendezvous to a small set of users' SmartPhones that were equipped with Bluetooth radios and Wi-Fi wireless networking.

6.1. Connection-Action Phrasing in BlueRendezvous

Stitching interactions are not possible on mobile phones that lack touch screens. BlueRendezvous is based on the simultaneous button press approach of SyncTap (Rekimoto, 2004), but requires users to press the same number on the keypad of their phones at the same time while running the connection screen of the BlueRendezvous

application. Hence, spurious connections would not occur while dialing a phone number, for example.

We designed our gesture recognition module to be flexible so that two cooperating users can press the same numeric key on their own phones at approximately the same time. This is in direct contrast to SyncTap's behavior, which requires one user to press a button on both devices in very tight synchrony. While straightforward, requiring one user to touch both devices at the same time to accomplish such a gesture stands in contrast to a phone's private nature, and its location in the intimate category of its owner's proxemic space.

BlueRendezvous also moves beyond SyncTap by demonstrating how devices with limited input capabilities can support richer connection-action phrases. Blue Rendezvous supports connections formed for the purpose of sending images captured from the camera phone to another user, sending a specific contact to another user, exchanging files, or performing a mutual business contact swap.

We treat the ubiquitous 12-key numeric keypad of a mobile phone as a very low-resolution touchpad. For example, ZoneZoom (Robbins, Cutrell, Sarin, & Horvitz, 2004) uses a smartphone's keypad as an input surface to pan and zoom through maps. Fathumb (Karlson, Robertson, Robbins, Czerwinski, & Smith, 2006), enables the navigation of hierarchical metadata with a similar approach. BlueRendezvous employs the cooperative style of synchronous gestures where each user performs their own part of the gesture. Thus, no violation of the other user's personal space is necessary to perform the gesture.

To link two devices using BlueRendezvous, each user runs the application on their phone. The BlueRendezvous application automatically activates the Bluetooth radio for the user. A BlueRendezvous connection is established through the following steps:

1. Initiate. One user initiates the connection by pressing the *Find Partner* soft key. This identifies the initiating device as the sender.

2. Synchronize. Each user then must press the same numeric key (0-9) at about the same time (Figure 15). Social protocol takes an important part of this step, as users need to coordinate (i.e., agree) which key(s) to press, and when. This makes it less likely that collisions with other devices will occur, since there is not a single "sync" button, but rather the equivalent of a 9-channel walkie-talkie. It would be straightforward to extend BlueRendezvous to allow multi-digit codes, analogous to a PIN code for personal banking, but we have not yet implemented this.

3. Connect. After the synchronization, the sender uses the Bluetooth discovery protocol to identify other devices providing the BlueRendezvous service. Only a BlueRendezvous service that corresponds to the numeric key press selected in step 2 will be discovered. In our present implementation, all communication occurs over Bluetooth, but as Wi-Fi enabled phones become more common, Bluetooth could be used to synchronize devices and then hand off the connection to the Wi-Fi channel.

4. Act. Once the devices have established a connection, an action screen (Figure 15) that assigns a connection task to each numeric key on the keypad appears. Either user may now initiate any of the actions.

5. Right of Refusal: If one user initiates an action, then the other user is offered the opportunity to accept or refuse the results of that action. For example, when transferring images, the receiver sees a thumbnail of the image and is asked, "Do you want to receive this image?" before the entire image is downloaded.

6. Disconnect. Users can exit the action screen to close the connection. This automatically restores the Bluetooth radio to its previous state. This behavior saves power and ensures that the user's device is not accidentally left in the "discoverable" state.

Figure 15 ABOUT HERE

6.2. Details of the Synchronization / Connection Process

During the synchronization phase (step 2), the Bluetooth devices transmit signals to discover each other. During this discovery process, one device must transmit that it is 'discoverable' while the other device must listen for discoverable devices. A limitation of Bluetooth is that a device cannot perform both of these input/output functions at the same time. Because Bluetooth discovery typically takes 10 seconds to occur, the connection step is not as rapid as we would like. In our implementation, the initiating user can press the *Find Partner* softkey to identify his device as the discoverer and speed up this process. Alternatively, if users can forego this speedup action, each device randomly alternates between transmitting and listening until the devices find one another, but this may take up to a minute to occur. Another mechanism to disambiguate the roles of client and server devices is the use of a tap-and-hold gesture, which a user performs instead of a single tap on the device that she wants to assign the server role (Figure 16).

BlueRendezvous only requires the numeric key press on each device to be close enough in time so that one device has the opportunity to discover the other device before it stops transmitting.

Figure 16 ABOUT HERE

The synchronous gesture that defines a BlueRendezvous connection allows for a straightforward discovery and connection of smart phones. An important feature of this discovery and connection process is that even though the synchronous button presses provides minimal information about how to connect the devices, we structure the interaction in such a way that the initiation, connection, and action selection are all provided and incorporated into the task workflow. BlueRendezvous currently does not include any mechanism to specify the relative spatial locations of the devices. For operations where this information could be used to augment interactions across the connection, this could be provided via the numeric keypad, but this feature has not yet been added to the system.

6.3. Social Implications of BlueRendezvous

Unlike SyncTap (Rekimoto, 2004), BlueRendezvous does not require a carefully timed synchronous button press because a number of design properties filter out "false positive" connections. Firstly, the BlueRendezvous application does not monitor incidental keypresses: users must activate the BlueRendezvous application to signal that they desire to participate in a connection. Secondly, users must press the same key from among 10 possibilities. Thirdly, the limited range of the Bluetooth radio automatically limits the interaction to nearby discoverable devices, which in turn must offer the

BlueRendezvous service. Finally, users can fall back on social protocols such as verbal communication to verify that their devices have correctly connected as expected.

BlueRendezvous provides a direct solution for interactions in the social-to-public range of proxemic distance, as it naturally lends itself to forming connections without having to touch or even look at the screen of another person's device. In principle, since BlueRendezvous is an example of a cooperative synchronous gesture, the interaction could also support one-to-many connections, but this is not easy to implement using the current capabilities of Bluetooth devices. BlueRendezvous' reliance on social protocol implicitly enhances the security of the technique, since communication of when to synchronize, and what key to press, occurs on a different communication channel. Establishing a secure communications channel is not the focus of our efforts, although this issue is discussed in more depth by others (Rekimoto, J., 2004; Rekimoto, J., et al., 2003). This is one area where NFC or physical transport strategies ("Near Field Communications," 2008; Want, Fishkin, Gujar, & Harrison, 1999; Zimmerman, 1995) have a distinct advantage, because a cryptographic key can be exchanged in this manner with limited possibility of eavesdropping by another person or device.

Because BlueRendezvous facilitates one-to-one connections, once a connection is established, the feedback on the screens makes it clear whether or not the intended devices were actually connected. A "man in the middle" device would not be able to connect at the same time as the intended device. The right of refusal option also gives users the opportunity to back out of an unwanted connection or to cancel the reception of undesired information. We have considered supporting an option to make refusals silent to the partner device, so as to ease social tensions if a user decides that they really don't

want to accept a file or connection that another user offers – without causing the other user to lose face.

7. BLUETABLE

BlueTable (Wilson & Sarin, 2007) is an integration of BlueRendezvous with the PlayAnywhere system (Wilson, 2005) that demonstrates how an interactive horizontal display with an integrated sensing system can enable novel interactions between a collaborative horizontal display and mobile devices that users bring to a meeting. The recently announced Microsoft Surface Computer (Figure 17) demonstrates similar interactions, based on BlueTable, to allow transfer and interactive manipulation of tracks between Zune music player devices with wireless networking capabilities.

Figure 17 ABOUT HERE

BlueTable integrates a short-throw projector and a vision-based sensing module in a compact footprint. This enables interactive applications on a horizontal surfaces of up to 40 inches diagonal (Figure 18). The sensing module uses an infrared technology that can perceive hand gestures on the surface as well as the placement of physical objects on the surface. The overall system is capable of juxtaposing graphics with the sensed objects on the surface. With this hardware configuration, BlueTable offers a rich connection infrastructure between Bluetooth-enabled mobile devices that are placed on the table, and the large horizontal display itself, while adding rich multi-touch hand gesture capabilities.

Figure 18 ABOUT HERE

BlueTable facilitates the connection of mobile devices through the following series of steps. When the system detects a phone-like shape over its display surface (Figure 19.1), it tells in-rage Bluetooth devices (Figure 19.2) to generate a signal in the visible or IR spectrum (Figure 19.3); if the table's vision system sees the signal from the detected device's location, a connection is established. Thus, the synchrony of signals across multiple devices again serves to establish communication between devices.

Immediately after the connection is established, information such as recent photos taken on a camera phone "spill" out of the mobile device (Figure 19.4). Because of the interactive nature of the BlueTable, users can interact with this information by touching table's surface (Figure 19.5). Users can inspect, manipulate, modify or share the photos with other devices already on the table, by taking advantage of a rich multi-touch, high-resolution interactive surface, instead of restricting all interactions to the limited set of mobile phone inputs.

Figure 19 ABOUT HERE

The connection metaphor for BlueTable is simple and has powerful interaction affordances. The act of physically resting one's device on a surface to establish a connection is an unmistakable physical act that users immediately understand. This physical act also is immediately visible to all co-located users in a manner that synchronous button presses or pen gestures may not be. The horizontal form factor of the table is crucial to this metaphor. Gravity holds the devices on the surface and keeps a connection live. The user can remove the devices from the surface to pull away from the connection. Other users can add their devices to the surface to participate or bring new

content into the interaction. Putting one's mobile on the surface transitions the device from a personal, private role to a public, shared role in a rapid and intuitive manner, yet picking the device back up immediately reverts it to a private display.

BlueTable provides a lucid example of how combinations of devices with dissimilar roles and capabilities can lead to novel interaction possibilities. The table, by itself, is not good for presenting private information, and the phone, by itself, offers very limited inputs and does not afford collaborative interactions. Together, the devices provide a rich set of possibilities with fluid transitions between private and public roles for information and interaction.

8. DISCUSSION

While different in their implementation and form, the projects and interactions we have presented in this article share the same common vision of linking multiple devices into a heterogeneous sensing/display environment using synchronous gestures. These projects led us to understand the importance of considering proxemics as a guiding principle for the design of cross-device interactions. Furthermore, our prototypes, studies, and observations suggest important lessons for desirable design properties of multi-device and multi-user connection phrases.

8.1. Support for Actions at Different Proxemic Distances

Because of the nature of ad-hoc, multi-display and device environments, one should allow for interaction designs supporting different device and body orientations. Considering the social expectations of proxemics can reveal important functional requirements, which help designers shape effective solutions during interaction design

processes. For example, even at intimate settings, physical contact between devices is likely to be acceptable, but one cannot make the same assumption for interaction at further distances.

Figure 20 illustrates how our different projects relate in terms of the proxemic distances in which they can occur. The appropriateness of a connection-action interaction is not only dependent of the range of proxemic distances in which it occurs, since other factors can influence its suitability for a task or scenario. The amount of information encoded in the connection phrase is important as well. For example, bumping provides a straightforward mechanism to create shared workspaces at intimate distances, yet the geometry information this connection mechanism carries is coarse. Stitching allows for the specification of more precise geometry data, yet it does not provide a disconnection metaphor with a strong affordance as bumping does. The BlueTable project seems to address both these issues: the support of intimate distances and a reversible connection / disconnection metaphor, still this solution does not scale well for purely mobile scenarios.

Figure 20 ABOUT HERE

The ability to gather geometry and location information during a connection phrase is important at close distances where it can assist in providing strong and meaningful feedback for shared workspaces or information exchanges. Techniques such as bumping, stitching and systems such as the BlueTable provide examples of the use of this geometry information. Cooperative stitching does not capture precise geometry information in the case of one-to-many connections, thus precluding us exploring the use of this type

information. Still, cooperative stitching can be used at a variety of proxemic distances and physical orientations, making it a flexible connection mechanism. This breath is due in great part to a balance between social protocol, which allows for coordination at-a-distance, and adequate graphical feedback, which brings awareness to the participating users.

8.2. Workflow

One should consider designs that fit into or that become part of a task's workflow. For example if part of the process of sharing a document involves getting physically close to a co-worker, it would be advantageous to use the act of approaching two devices together as part of the connection mechanism. Similarly, one can model a connection phrase using a gift-giving metaphor, where one offers copies of a document to a group of colleagues by holding the documents on one's hand. These two examples correspond to the Bumping and Stitch+Hold techniques. BlueTable provides a similar workflow integration by letting users perform straightforward and tangible direct manipulations that match users' expectations.

The structure of an interaction phrase plays a role in fitting a technique into a task's workflow. An examination of an interaction phrase's syntax can help not only designing a better interaction technique, but also discovering problems with existing ones. For example, the "establish and relax" pattern of activity we observed during intimate distances offers an argument against the use of post command menus, (Figure 5, lower-right). A lesson from this observation is to consider command specification options occurring on the initiating stitching device – e.g. prefix menus or postfix menus where a user returns to his local device.

The integration of an interaction technique into a task's workflow is one of the main differences between the projects we presented and the related work. For example, Pick and Drop focuses on the connection step and a default action, while stitching, supports the user's overall workflow, which permits several operations. The BlueTable is an interesting example of integration where the device *is* the medium where task occurs, thus making it a defining part of the workflow. BlueRendezvous requires a series of discrete steps to complete a useful operation, but while this process is less fluid than interactions such as stitching, it is technique that constitutes a step in the right direction and an improvement over current mobile phones discovery and connection mechanisms.

8.3. Privacy and Security

One should provide people participating in an ad-hoc connections with privacy and security mechanisms that let them exercise control over incoming and outgoing connections and data. This is an important principle that must be balanced against the previous principles of workflow and proxemic distance – i.e., a control mechanism should integrate into a task's workflow and be applicable at the appropriate proxemic distance. Cooperative stitching explicitly includes in its design features that have these properties, e.g., in the form of crossing widget interactions. Other connection techniques include security features that correspond to the metaphor that inspires them. In the case of bumping, separating device is a straightforward phrase for breaking a connection. BlueTable provides a similar privacy control mechanism – i.e., withdrawing a device from the table breaks its connection. These last two privacy mechanisms integrate gracefully into their respective connection phrases; yet they provide a coarse, absolute control over a connection and the information passing through it. We envision security

mechanisms, similar to the ones illustrated by Wu (Wu & Balakrishnan, 2003) in the context of tabletop displays, that does not require completely severing a connection.

Our observations revealed that people's sense of property and ownership could play an important factor in the design of connection phrases and metaphors. For example, an intermediary device that provides a "neutral territory" can alleviate concerns about close proximity to strangers – e.g., portal or candy-dish metaphors (Hinckley, Ramos, Guimbretiere, Baudisch, & Smith, 2004). If not dedicated, personal devices may need a "media container" mode. However, the owner of such a device may not feel comfortable having co-workers grab a device that might contain personal information. We received this type of comments from users during our early cooperative stitching design sessions. Some devices have no owners, e.g., BlueTable is an active surface that mediates between the devices people put over it and acts as an intermediary, neutral entity that lessens property concerns for interactions at close distances. BlueRendezvous mode of interaction is at the other end of the spectrum, where users keep their respective devices to themselves, and using social protocol to mediate a connection.

8.4. Eight Design Questions for Ad-Hoc Multi Device Environments

The previous discourse prompts us to view and consider connection mechanisms not only in the context of proxemics, but also on fundamental interaction issues. We summarize these issues in eight design questions for connection-action phrases for ad-hoc multi-device environments.

1. **Connection:** How is a connection established?

2. **Command**: What type of connection is required? What is the connection used for?

3. **Operands**: What information is shared?

4. **Geometry**: What is the spatial relationship and orientation between devices?

5. **Coexistence**: How does a connection mechanism coexist with traditional interactions or naturally occurring user behaviors?

6. **Proxemics**: Does a connection mechanism take into consideration or affect users' share of the physical space?

7. **Privacy:** How does a user avoid undesired connections? Can the user control what content is shared? How does either user back out of an operation at any point, or veto a connection that is no longer desired?

8. **One vs. Many Partners:** As made apparent by our explorations of various synchronous gestures, a given ad-hoc connection technique may not support the requirements of one-to-one and one-to-many connections equally well. The designer must decide which types of collaborations are most important to support in a given application or user task scenario.

We have explored a number of point designs that offer several ways to address these questions, but many additional approaches or creative hybrid solutions that combine the properties of different approaches remain to be explored. We believe that the concept of connection-action phrases can help to encourage novel cross-device connection mechanisms that fit into the flow of other user interactions.

9. FUTURE WORK

Our research represents a step towards post-WIMP user interfaces in the context of ad-hoc gatherings of potentially connected displays and devices. Still there is much yet to explore. For example, while we had the opportunity to test important aspects of our cooperative designs, there are still areas that require quantitative evaluations. In future studies we would like to explore the different stitching techniques, particularly to quantitatively analyze Stitch+Lift and Stitch+Hold gestures with intermediate feedback. While users received well the privacy mechanisms we have introduced, we also need to formally observe and evaluate them.

Future work also includes supporting users that may not want to have to respond to every incoming connection they receive within the time-span of a particular synchronous gesture. An asynchronous stitching interaction that combines broadcasting with a mailbox metaphor is a design that could address this issue (Figure 21). For example, messages or incoming connections from a sender device could accumulate on an inbox space at the receiving devices. Later, users can consider if and how they wish to handle these requests. This idea relates to the Sharing Palette file-sharing system (Voida, Edwards, Newman, Grinter, & Ducheneaut, 2006), which allows for an elegant way of specifying user groups, permissions and sharing styles. Stitching is a connection mechanism that could assist in rapidly populating the palette's list of available users.

Figure 21 ABOUT HERE

There are interesting aspects of BlueRendezvous that we plan to explore. BlueRendezvous provides users with the means to establish a connection between

devices only requiring a keypad and a compatible wireless link. These requirements could make this connection technique suitable for a number of devices beyond mobile phones. For example, richer devices such as tablets or large interactive displays can provide users with access to virtual keypads, if necessary.

Connecting to a nearby device is only part of a BlueRendezvous session, which also requires the use of action screens in order for a particular activity to take place. We envision action screens reflecting the idiosyncrasies of the devices on which they appear – e.g., mobile phones, digital whiteboards, etc. Action screens provide us with the opportunity to explore additional functionality that we did not implement on the deployed BlueRendezvous system. For example, we can use action screens to provide geometry information about a connection so that a user could annex a nearby device's display. Figure 22 shows a mockup design of an action screen with keys that allow a user to specify the relative orientation of the pairing device and show a shared image across two screens.

Figure 22 ABOUT HERE

In this example, a user selecting the share option for a particular photo can immediately add a geometry operand; in this case, the user presses the "6" key. The receiving device notifies its user of an incoming share operation and provides the option to add geometry information to it; in this case, the receiving user presses the "4" key. Lead by social protocol, users can establish a side-by-side connection, which allows the devices to share the image across their screens, similarly to what bumping and stitching permit.

10. CONCLUSION

In this article, we focus on the following research question: how can users dynamically forge a purposeful connection between two or more target devices that do not know a priori one another's information? This question reveals pressing research problems for system architecture, interaction design, as well as social and behavioral observation studies. We explore ways of combining two or more devices into a heterogeneous display environment in an interactive, ad-hoc fashion, such as during a meeting. In particular, we focus on the use of synchronous gestures to achieve spontaneous connections between collocated displays and devices.

We present a number of recent and novel technologies, interaction designs and implementations that leverage the concept of synchronous gestures. At the same time, we present a design approach that considers cross-device operations in general, including the system, interaction, and social issues that arise. The related literature presently lacks such a discussion of general requirements, making our discourse a concrete contribution of our research.

It is currently difficult to grasp what it will mean for everything and everyone to be wirelessly connected. What we are sure of is that some of the most exciting examples of post-WIMP user interfaces might arise from this primordial mix of wireless networking, heterogeneous mobile devices with disparate capabilities and ubiquitous interactive displays. The research we present in this article is an initial attempt to explore what might be possible and constitute a step towards this exiting research direction.

NOTES

Background.

Acknowledgments.

Support.

Authors' Present Addresses.

Gonzalo Ramos

136 – 102nd Ave SE, Bellevue, WA, USA, 98004

bonzo@dgp.toronto.edu

Kenneth Hinckley

Microsoft Research - One Microsoft Way, Redmond, WA U.S.A. 98052

kenh@microsoft.com

Andy Wilson

Microsoft Research - One Microsoft Way, Redmond, WA U.S.A. 98052

awilson@microsoft.com

Raman Sarin

Microsoft Research - One Microsoft Way, Redmond, WA U.S.A. 98052

ramans@microsoft.com

HCI Editorial Record. (supplied by Editor)

REFERENCES

Accot, J., & Zhai, S. (2002). More Than Dotting the i's - Foundations for Crossing-Based Interfaces. ***Proceedings of the CHI '02 Conference on Human Factors in Computer Systems,*** **New York, NY, 73-80.**

Apitz, G., & Guimbretière, F. (2004). Crossy: A Crossing-Based Drawing Application. ***Proceedings of the UIST '04 Symposium on User Interface Software and Technology,*** **Santa Fe, USA, 3-12.**

Ayatsuka, Y., & Rekimoto, J. (2005). Transticks: Physically Manipulatable Virtual Connections ***Proceedings of the CHI '05 Conference on Human Factors in Computing Systems*** **Portland, USA, 251-260.**

Buxton, W. (1986). Chuncking and Phrasing and the Design of Human-Computer Dialogues. In Baecker, Grudin, Buxton & Greenberg (Eds.), ***Readings in Human-Computer Interaction: Towards the Year 2000*** **(pp. 494-499). San Fransisco, CA: Morgan Kaufmann.**

Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., & Balakrishnan, R. (2006). Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices. ***Proceedings of the CHI '06 Conference on Human Factors in Computing Systems,*** **Montreal, Quebec, Canada, 861-870.**

Hall, E. T. (1966). ***The Hidden Dimension*****. Garden City, N.Y.: Doubleday.**

Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G., & Krohn, A. (2005). A Relative Positioning System for Co-Located Mobile Devices. ***Proceedings of the 3rd***

International Conference on Mobile Systems, Aapplications, and Services, Seattle, Washington, 177-190.

Hinckley, K. (2003). Synchronous Gestures for Multiple Persons and Computers. *Proceedings of the UIST '03 Symposium on User Interface Software and Technology*, Vancouver, Canada, 149-158.

Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., & Smith, M. (2004). Stitching: Pen Gestures That Span Multiple Displays. *Proceedings of the AVI'04 International Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 23-31.

Holmquist, L. E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., & Gellersen, H. (2001). Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. *Proceedings of the UbiComp '01 International Conference in Ubiquitous Computing*, Atlanta, USA, 116-122.

Karlson, A. K., Robertson, G. G., Robbins, D. C., Czerwinski, M. P., & Smith, G. R. (2006). Fathumb: A Facet-Based Interface for Mobile Search. *Proceedings of the CHI '06 Conference on Human Factors in Computing Systems*, Montreal, Canada, 711-720.

Kurtenbach, G., & Buxton, W. (1993). The Limits of Expert Performance Using Hierarchical Marking Menus. *Proceedings of the CHI '93 Conference on Human Factors in Computing Systems*, New York, USA, 35-42.

Mir, N. F. (2006). *Computer and Communication Networks*: Prentice Hall.

Morris, M. R., Huang, A., Paepcke, A., & Winograd, T. (2006). Cooperative Gestures: Multi-User Gestural Interactions for Co-Located Groupware. ***Proceedings of the CHI '06 Conference on Human Factors in Computing Systems*****, Montreal, Canada, 1201-1210.**

Near Field Communications. (2008). 2008, from http://en.wikipedia.org/wiki/Near_Field_Communication

Rekimoto, J. (1997). Pick and Drop: A Direct Manipulation Technique for Multiple Computer Environments. ***Proceedings of the UIST '97 Symposium on User Interface Software and Technology*****, New York, USA, 31-39.**

Rekimoto, J. (2004). Synctap: Synchronous User Operation for Spontaneous Network Connection. ***Personal UbiComp., 8*****(2), 126-134.**

Robbins, D. C., Cutrell, E., Sarin, R., & Horvitz, E. (2004). Zonezoom: Map Navigation for Smartphones with Recursive View Segmentation. ***Proceedings of the AVI'04 International Working Conference on Advanced Visual Interfaces*****, Gallipoli , Italy.**

Rodden, T., Rogers, Y., Halloran, J., & Taylor, I. (2003). Designing Novel Interactional Workspaces to Support Face to Face Consultations. ***Proceedings of the CHI '03 Conference on Human Factors in Computing Systems*****, Ft. Lauderdale, USA, 57-64.**

Scott, S. D. (2003). Territory-Based Interaction Techniques for Tabletop Collaboration. ***Proceedings of the UIST '03 Symposium on User Interface Software and Technology Companion*****, 17-20.**

Scott, S. D., Grant, K. D., & Mandryk, R. L. (2003). System Guidelines for Co-Located, Collaborative Work on a Tabletop Display. ***Proceedings of the ECSCW'03 European Conference Computer-Supported Cooperative Work*****, Helsinki, Finland, 159-178.**

Voida, S., Edwards, W. K., Newman, M. W., Grinter, R. E., & Ducheneaut, N. (2006). Share and Share Alike: Exploring the User Interface Affordances of File Sharing. ***Proceedings of the CHI '06 Conference on Human Factors in Computing Systems*****, Montrèal, Quèbec, Canada, 221-230.**

Want, R., Fishkin, K. P., Gujar, A., & Harrison, B. L. (1999). Bridging Physical and Virtual Worlds with Electronic Tags. ***Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*****, 370-377.**

Wilson, A. (2005). Playanywhere: A Compact Interactive Tabletop Projection-Vision System. ***Proceedings of the UIST '05 Symposium on User Interface Software and Technology*****, Seattle, USA, 83-92.**

Wilson, A., & Sarin, R. (2007). Bluetable: Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-Based Handshaking. ***Proceedings of the GI '07 Graphics Interface Conference*****, Montreal, Canada.**

Wu, M., & Balakrishnan, R. (2003). Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. ***Proceedings of the UIST '03 Symposium on User Interface Software and Technology,*** **Vancouver, Canada, 193-202.**

Zimmerman, T. G. (1995). ***Personal Area Networks (Pan): Near-Field Intra-Body Communication.*** **MIT, Cambridge.**

FOOTNOTES

(Make a copy of all footnotes on a separate page here. This only has to be done for the final submission for production. During the review process, it is okay to just have footnotes at the bottom of pages.)

1 Personal communication, Geoffrey T. Raymond, Assistant Professor of Sociology, University of California Santa Barbara.

FIGURE CAPTIONS

Figure 1: The four proxemic distances.

Figure 2: (Left) A user can bump a tablet into another one resting on a desk. The software recognizes the gesture by synchronizing the two accelerometers across a wireless network. (Right) The tablet moved by the user annexes the display of the stationary tablet, allowing a panoramic image to span both displays.

Figure 3: Top/down and left/right accelerometer signatures for bumping a left side of tablet onto the right side of another. Each tablet experiences a roughly equal but opposite pattern of forces. The left/right accelerometer axes exhibit characteristic spikes resulting from equal and opposite contact forces. Note the top/down axes exhibit no significant response.

Figure 4: Bumping as a synchronous gesture step-by-step. 1- A user bumps a tablet against another. 2- Registered devices stream sensor data to a synchronous server. 3- Server determines that the signals from devices A and B form a synchronous gesture (device C does not participate). 4- Server transmits to devices A and B information pertaining to whom they should connect. 5- Devices can communicate with each other and can continue exchanging additional information such as operation, commands, etc.

Figure 5: Connecting two tablets into a gallery configuration. (Top-Left) A user selects a photo thumbnail and performs a stitching gesture across two interactive screens (the red arrow is superimposed to illustrate the path the pen will follow).

(Top-Right) Once the pen reaches the neighboring screen, a post-gesture menu appears after a predetermined time-out. The user selects the "gallery option" by selecting/marking the adequate menu item (the red arrow illustrates the path the pen will follow to select the menu item). (Lower-Left) After the gallery configuration becomes active, the device on the left will "project" photos onto the device on the right. (Lower-Right) Detail of the post-gesture menu. The shadow-beams from a thumbnail to a menu or full-size photo take advantage of geometry calibration information and reinforce the notion of a united screen space.

Figure 6: (Left) Conceptual diagram of stitching recognition showing two envelopes occurring within a certain time threshold. (Right) Corresponding stitching gesture, used to move some photos from one device to another. The trajectory the pen follows not only serves to establish a connection, it also allows to provide geometry information that allows screens to be aligned if desired. The shadow trail spanning both screens takes advantage of this information in order to provide visual feedback.

Figure 7: Functionality enabled by stitching operations within the context of our photo-sharing prototype: StitchMaster

Figure 8: Device calibration through stitching. Stitching approximates the pen's path that traverses both devices by a straight line. This approximation not only fits real trajectories well, but also greatly simplifies the calibration process. First, the system estimates the intersection of the stitching gesture with the edges of each screen, yielding the points PA and p1 with incidence angles $\alpha 0$ and $\alpha 1$, respectively. The offset between the two screens is then tan(α) times the combined bezels width (B1+B2), where α stands for the average of $\alpha 0$ and $\alpha 1$. This angle is a good estimator of the incidence angle of the line between PA and p1. Using the previous information the system can compute PB as the displacement of p1 along the edge of Device #2's screen *by offset* pixels. With this information, the system can transform points from one device's coordinate system to the other, thus allowing the presentation of graphics that appear to span both devices. These transformation functions $Fi_{,j}$: R2 $^{\rightarrow}$ R2 from the *i* to the *j* device's coordinate systems are F1,2(x,y) = (- (B1 + B2 + *screen_width1*) + x, *offset* + y) and F2,1(x,y) =(B1 + B2 + *screen_width1* + x, *offset* + y). In order for the calibration to occur, parameters such a device's bezel width and screen size (such as *screen_width*) are well-know magnitudes that the stitching arbiter receives when the devices connects to the arbiter for the first time.

Figure 9: In the general case, stitching occurs between devices positioned in one of 16 possible configurations. Using rotation transforms T_i: $R^2 \rightarrow R^2$, we can change the

devices' coordinate system such that it seems that a stitching operation always occurs from a leftmost device to a rightmost device. This transformation allows us to consider only one calibration case. (Left) Two tablets are touching through their top edge and a stitching gesture is running from device #1 to device #2. (Right) Rotating device #1's coordinate system 90 degrees and device #2's coordinate system -90 degrees allows us to consider these devices as if they were in a well-know configuration for which there is a well-know calibration solution.

Figure 10: Our stitching implementation on a pocket PC. (Left) A user drags a thumbnail from the device on the left until the stylus reaches the edge of the screen. Then the user resumes the (dragging) stroke from the edge of the device on the right. (Right) after the stitching gesture, the system provides visual feedback of the operation.

Figure 11: (Top Left) Stitch+Lift. The sender initiates the connection, and each recipient must accept the offer within a short time window. (Top Right) Stitch+Hold. The sender initiates the connection and holds the pen at the top of the screen until all desired recipients have accepted the connection. In this figure, marks with the same shade of gray are done by the same person. A multi-shaded stitch is done by two people, e.g. black-light gray. (Bottom) Stitching. To connect to two other devices, a sender (A) draws a stitch onto each device (B,C).

Figure 12: Either user (here, the sender on the right) can rescind a connection by simply crossing out the undesired user's icon. This action cancels the file transfer between these users.

Figure 13: Three users' devices arranged around a table. One user (A) initiates a transfer by dragging files to screen edge. Other co-located users (B, C) can receive them by pulling down from the top of their screens. As users B and C start dragging, intermediate feedback identifies the sender and previews the files IT offers. The sender (A) then sees feedback showing who has accepted the proposed connection.

Figure 14: Arrangement of devices used in our study.

Figure 15: (Left) A diagram depicting a BlueRendezvous gesture – i.e., two users touching an agreed upon keypad button. (Right) The Actions screen, a ZoneZoom screen that allows either user to choose a cross-device operation. The available operations depend on the capabilities and user-controlled configuration of each device.

Figure 16: Tap-and-hold serves as a mechanism to distinguish between the client and server devices. The shaded area around the "release" event indicates that there is a time tolerance – i.e., keys do not have to be release exactly at the same time for a connection to occur.

Figure 17: (Left) The Microsoft's Surface computer. (Right) An example of a surface computer application displaying information "bleeding out" from a mobile device once put over the interactive surface.

Figure 18: (Left) PlayAnywhere system consists of a NEC WT600 projector on a short pedestal, a camera with infrared pass filter and infrared LED illuminant.

(Upper right) Input image with phone; (Lower right) Binarized image of a cell phone with ellipse fitted over it.

Figure 19: (Left) Steps leading to a connection between a mobile device and the BlueTable. (1) A user puts a device on the table. (2) The table's vision system detects a phone-like object on its surface and broadcasts Bluetooth signals to nearby devices instructing them to blink a certain way. (3) The device blinks on the predetermined pattern and gets detected/seen by the table. (4) A connection is established and a default interface "spills out" of the device on the table. (5) After that, the user can interact with the presented interface. (Right) Mockup of a complex user interface (calendar application) that spills out of a mobile device onto the BlueTable's surface.

Figure 20: Our different connection mechanisms and the proxemic distances at which they can be performed.

Figure 21: Mockup diagram of a stitching solution that follows a mailbox metaphor.

Figure 22: Mockup of an action screen to specify geometry information. The combination of keys pressed may indicate spatial operations as well as the type of function. The system does not sense the relative spatial location of the devices: users hit the arrow keys to specify this.

FIGURES

(IT'S BEST TO PUT ONE FIGURE PER PAGE)

Figure 1: The four proxemic distances.

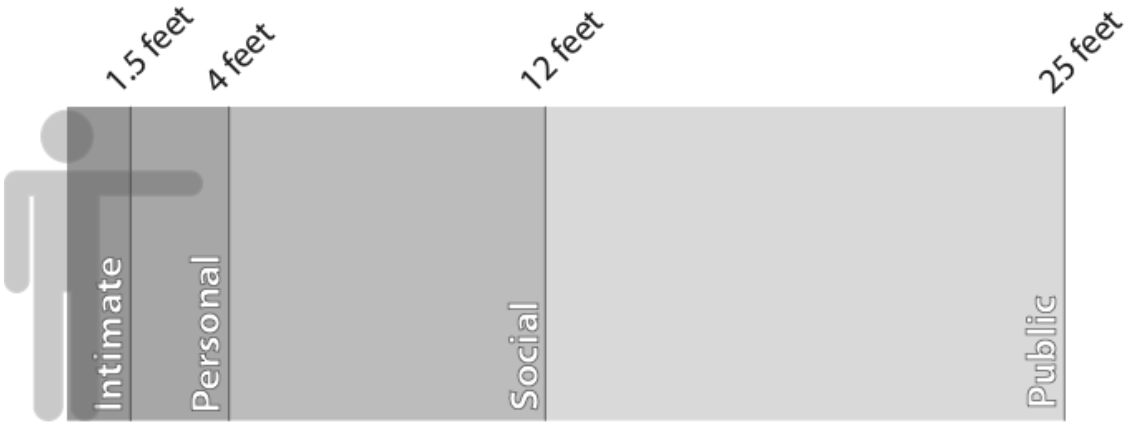

Figure 2: (Left) A user can bump a tablet into another one resting on a desk. The software recognizes the gesture by synchronizing the two accelerometers across a wireless network. (Right) The tablet moved by the user annexes the display of the stationary tablet, allowing a panoramic image to span both displays.

Figure 3: Top/down and left/right accelerometer signatures for bumping a left side of tablet onto the right side of another. Each tablet experiences a roughly equal but opposite pattern of forces. The left/right accelerometer axes exhibit characteristic spikes resulting from equal and opposite contact forces. Note the top/down axes exhibit no significant response.

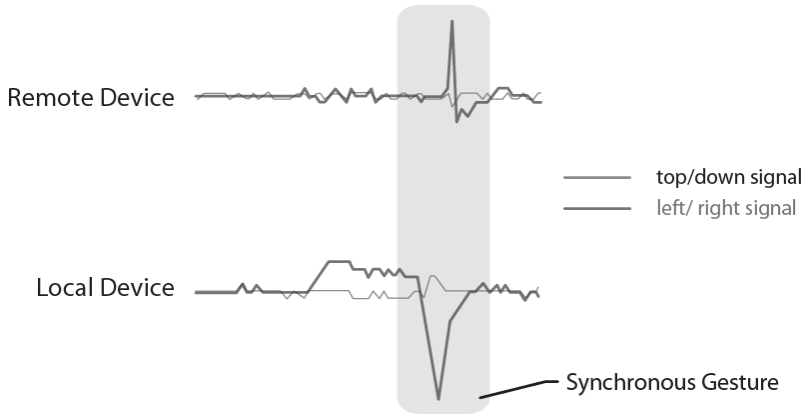

Figure 4: Bumping as a synchronous gesture step-by-step. 1- A user bumps a tablet against another. 2- Registered devices stream sensor data to a synchronous server. 3- Server determines that the signals from devices A and B form a synchronous gesture (device C does not participate). 4- Server transmits to devices A and B information pertaining to whom they should connect. 5- Devices can communicate with each other and can continue exchanging additional information such as operation, commands, etc.

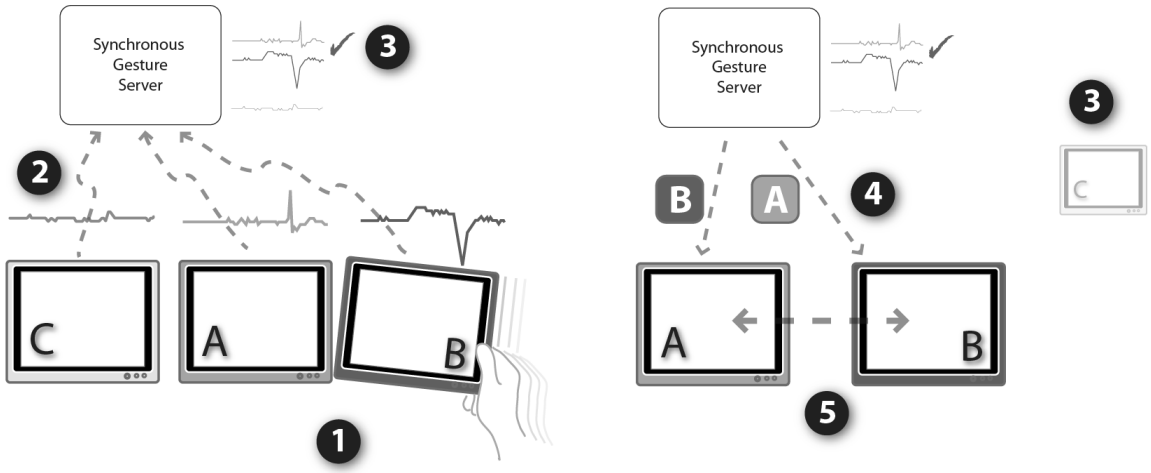

Figure 5: Connecting two tablets into a gallery configuration. (Top-Left) A user selects a photo thumbnail and performs a stitching gesture across two interactive screens (the red arrow is superimposed to illustrate the path the pen will follow). (Top-Right) Once the pen reaches the neighboring screen, a post-gesture menu appears after a predetermined time-out. The user selects the "gallery option" by selecting/marking the adequate menu item (the red arrow illustrates the path the pen will follow to select the menu item). (Lower-Left) After the gallery configuration becomes active, the device on the left will "project" photos onto the device on the right. (Lower-Right) Detail of the post-gesture menu. The shadow-beams from a thumbnail to a menu or full-size photo take advantage of geometry calibration information and reinforce the notion of a united screen space.

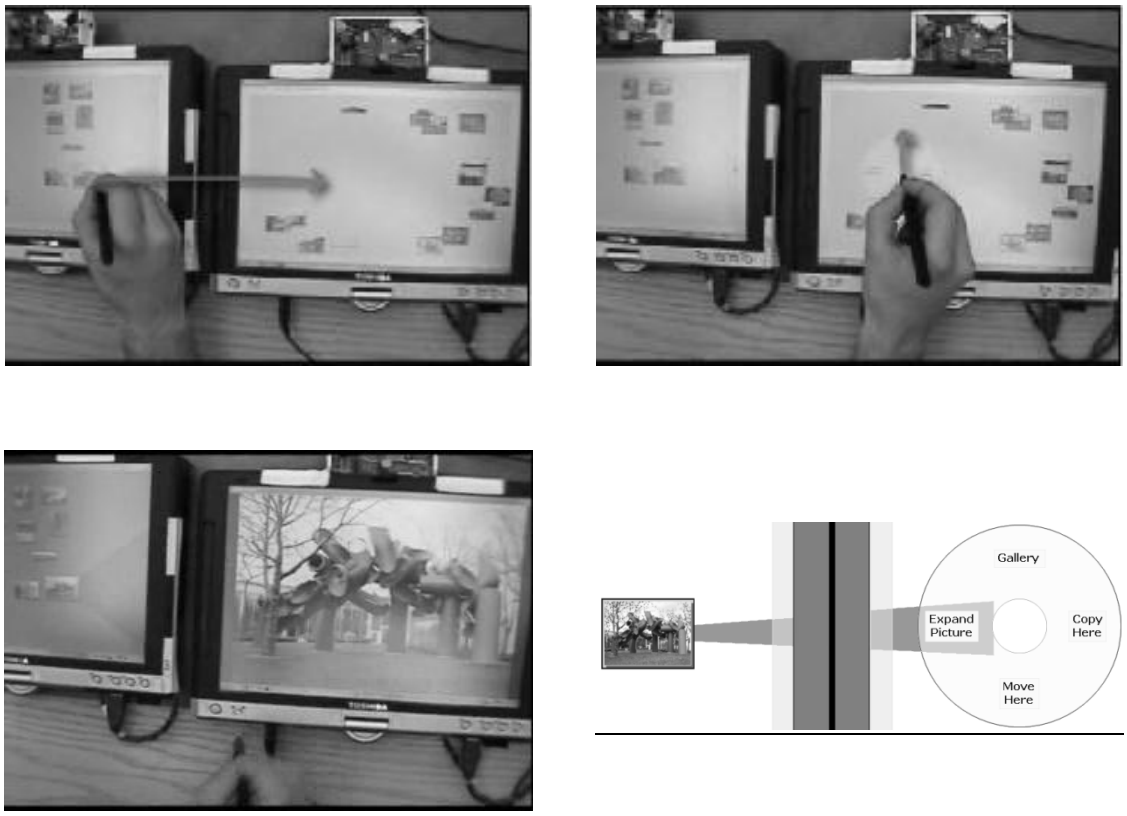

Figure 6: (Left) Conceptual diagram of stitching recognition showing two envelopes occurring within a certain time threshold. (Right) Corresponding stitching gesture, used to move some photos from one device to another. The trajectory the pen follows not only serves to establish a connection, it also allows to provide geometry information that allows screens to be aligned if desired. The shadow trail spanning both screens takes advantage of this information in order to provide visual feedback.

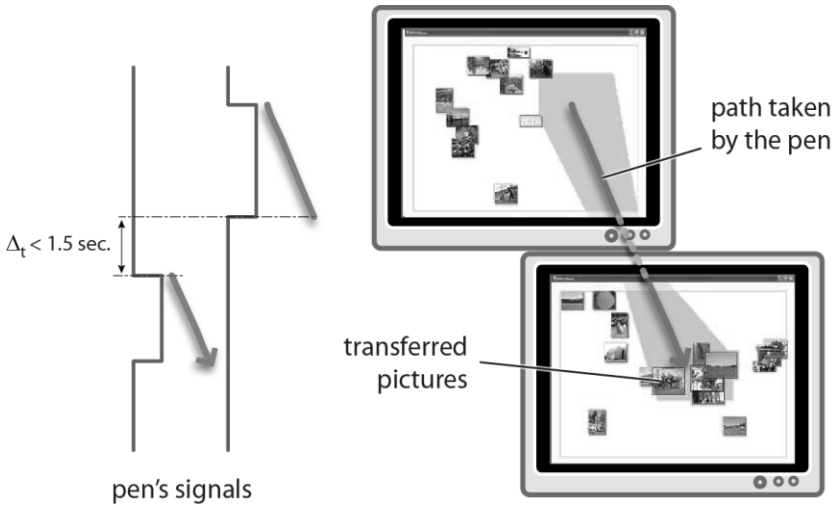

Figure 7: Functionality enabled by stitching operations within the context of our photo-sharing prototype: StitchMaster

Configuration /Action	Description	Stroke Starts @ ...	Post-Command
Move	Moves photo(s) from one device to the other.	Selected photo(s)	(default action)
Copy	Copies photo(s) from one device to the other.	Selected photo(s)	Yes
Gallery	The device where the stroke starts becomes a control and the device where the stroke ends acts as a photo display area. Selecting a thumbnail on the control device causes the display device to show it at full size.	Selected photo(s) / empty region	Yes
Expansion	A photo is maximized to occupy the expanded display region defined by tiling both participating devices.	Selected photo	Yes
Persistent Shared Workspace	The participating devices' screen are combined into a shared area with "transporter portals" that let users pass thumbnails without having to reach into another device's space.	Empty region	(default action)

Figure 8: Device calibration through stitching. Stitching approximates the pen's path that traverses both devices by a straight line. This approximation not only fits real trajectories well, but also greatly simplifies the calibration process. First, the system estimates the intersection of the stitching gesture with the edges of each screen, yielding the points P_A and p_1 with incidence angles α_0 and α_1, respectively. The offset between the two screens is then *tan*(α) times the combined bezels width (B_1+B_2), where α stands for the average of α_0 and α_1. This angle is a good estimator of the incidence angle of the line between P_A and p_1. Using the previous information the system can compute P_B as the displacement of p_1 along the edge of Device #2's screen by *offset* pixels. With this information, the system can transform points from one device's coordinate system to the other, thus allowing the presentation of graphics that appear to span both devices. These transformation functions $F_{i,j}$: $R^2 \rightarrow R^2$ from the *i* to the *j* device's coordinate systems are $F_{1,2}(x,y) = (- (B_1 + B_2 + screen\_width_1) + x, offset + y)$ and $F_{2,1}(x,y) =(B_1 + B_2 + screen\_width_1 + x, offset + y)$. In order for the calibration to occur, parameters such a device's bezel width and screen size (such as *screen_width*) are well-know magnitudes that the stitching arbiter receives when the devices connects to the arbiter for the first time.

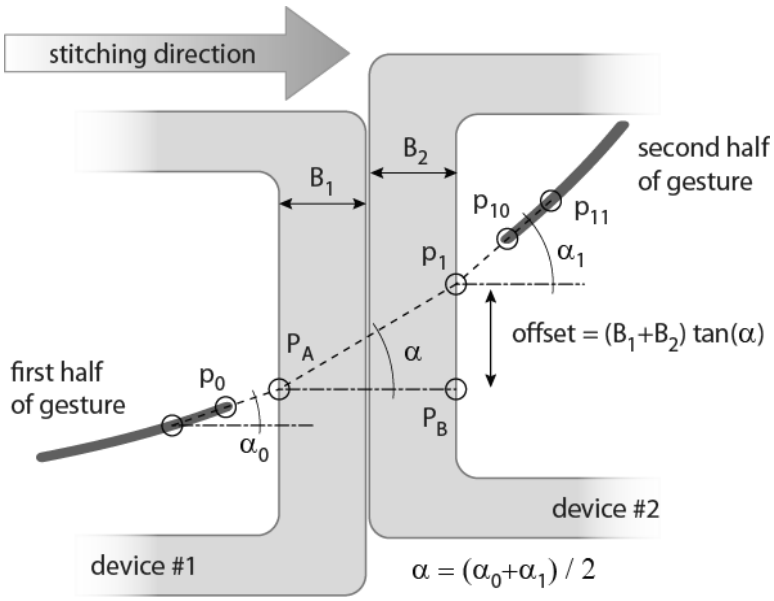
stitching direction
B1
B2
second half
of gesture
p10
p11
α1
p1
offset = (B1+B2) tan(α)
PA
α
first half
of gesture
p0
PB
α0
device #2
device #1
α = (α0+α1) / 2

Figure 9: In the general case, stitching occurs between devices positioned in one of 16 possible configurations. Using rotation transforms T_i: $R^2 \rightarrow R^2$, we can change the devices' coordinate system such that it seems that a stitching operation always occurs from a leftmost device to a rightmost device. This transformation allows us to consider only one calibration case. (Left) Two tablets are touching through their top edge and a stitching gesture is running from device #1 to device #2. (Right) Rotating device #1's coordinate system 90 degrees and device #2's coordinate system -90 degrees allows us to consider these devices as if they were in a well-know configuration for which there is a well-know calibration solution.

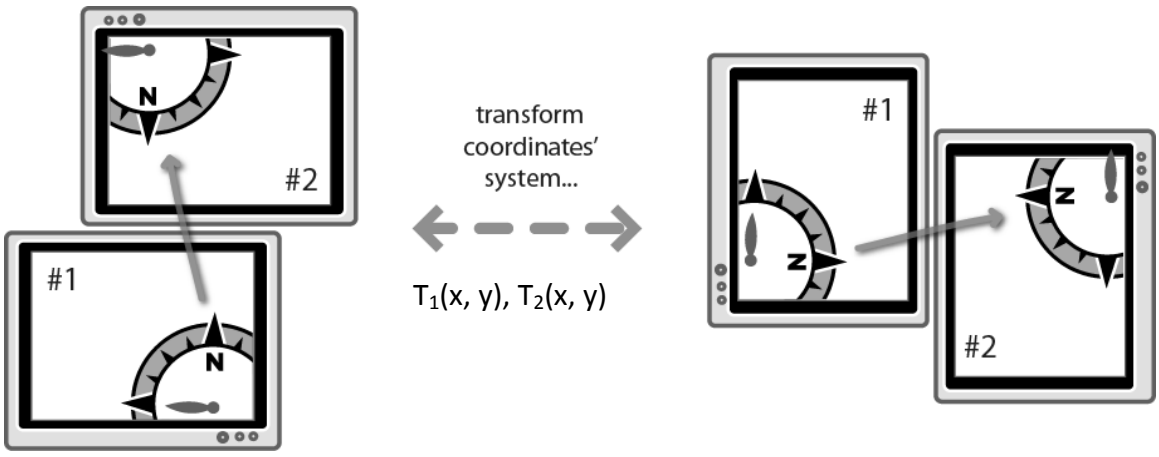

Figure 10: Our stitching implementation on a pocket PC. (Left) A user drags a thumbnail from the device on the left until the stylus reaches the edge of the screen. Then the user resumes the (dragging) stroke from the edge of the device on the right. (Right) after the stitching gesture, the system provides visual feedback of the operation.

Figure 11: (Top Left) Stitch+Lift. The sender initiates the connection, and each recipient must accept the offer within a short time window. (Top Right) Stitch+Hold. The sender initiates the connection and holds the pen at the top of the screen until all desired recipients have accepted the connection. In this figure, marks with the same shade of gray are done by the same person. A multi-shaded stitch is done by two people, e.g. black-light gray. (Bottom) Stitching. To connect to two other devices, a sender (A) draws a stitch onto each device (B,C).

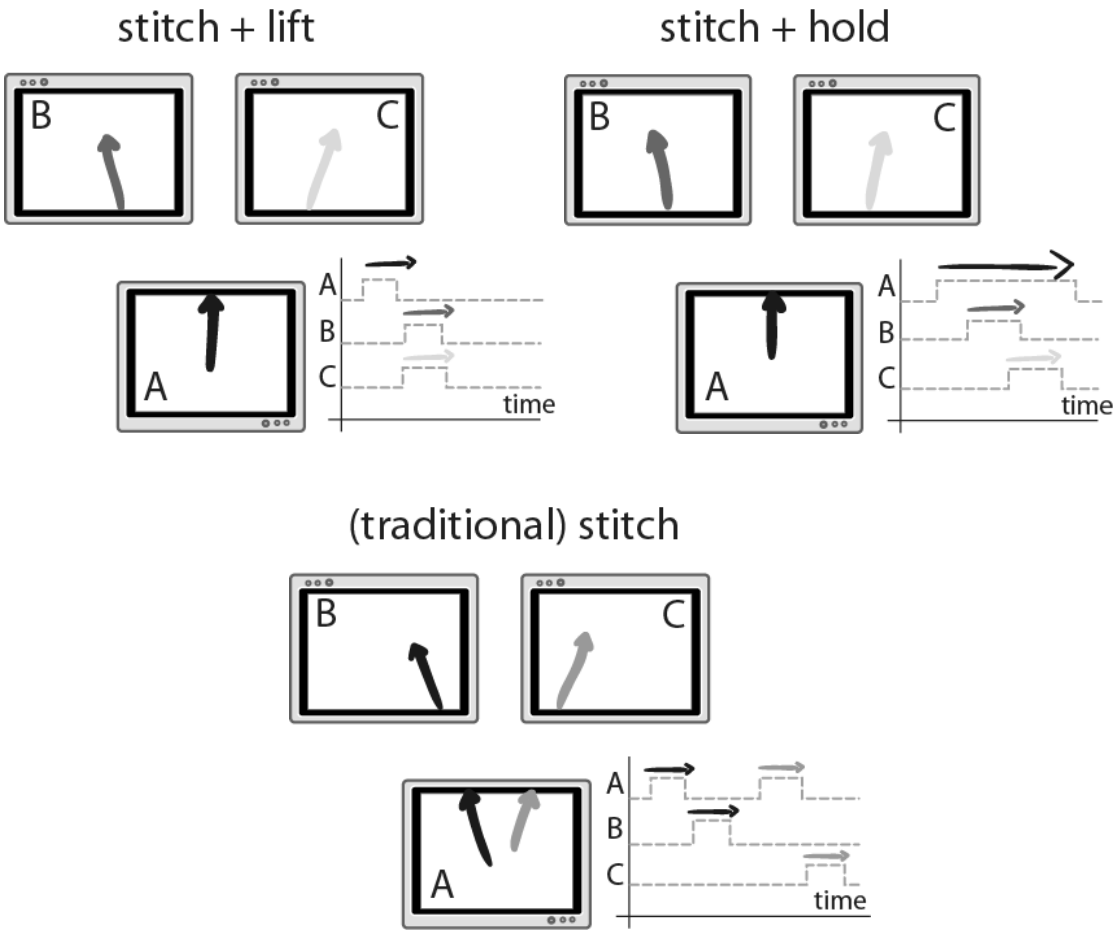

Figure 12: Either user (here, the sender on the right) can rescind a connection by simply crossing out the undesired user's icon. This action cancels the file transfer between these users.

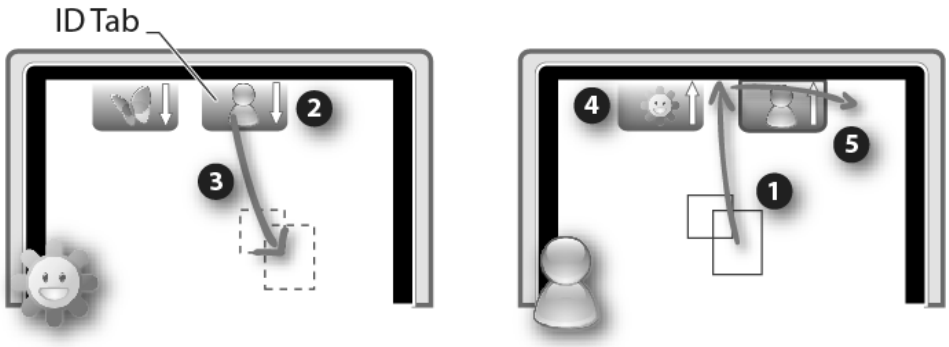

Figure 13: Three users' devices arranged around a table. One user (A) initiates a transfer by dragging files to screen edge. Other co-located users (B, C) can receive them by pulling down from the top of their screens. As users B and C start dragging, intermediate feedback identifies the sender and previews the files IT offers. The sender (A) then sees feedback showing who has accepted the proposed connection.

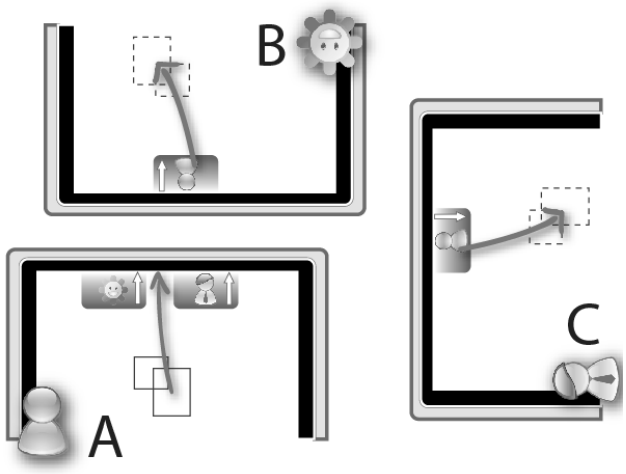

Figure 14: Arrangement of devices used in our study.

Figure 15: (Left) A diagram depicting a BlueRendezvous gesture – i.e., two users touching an agreed upon keypad button. (Right) The Actions screen, a ZoneZoom screen that allows either user to choose a cross-device operation. The available operations depend on the capabilities and user-controlled configuration of each device.

Figure 16: Tap-and-hold serves as a mechanism to distinguish between the client and server devices. The shaded area around the "release" event indicates that there is a time tolerance – i.e., keys do not have to be release exactly at the same time for a connection to occur.

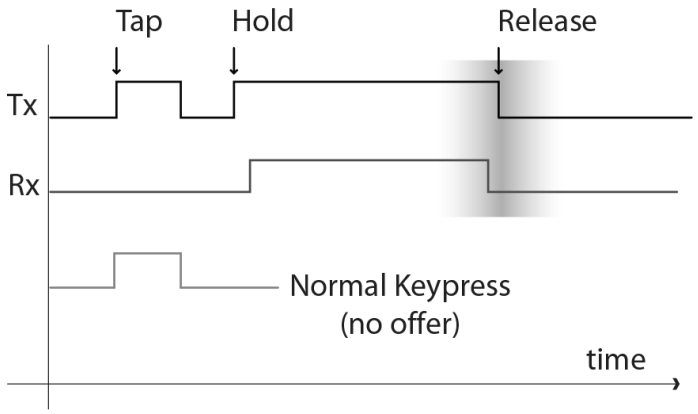

Figure 17: (Left) The Microsoft's Surface computer. (Right) An example of a surface computer application displaying information "bleeding out" from a mobile device once put over the interactive surface.

Figure 18: (Left) PlayAnywhere system consists of a NEC WT600 projector on a short pedestal, a camera with infrared pass filter and infrared LED illuminant. (Upper right) Input image with phone; (Lower right) Binarized image of a cell phone with ellipse fitted over it.

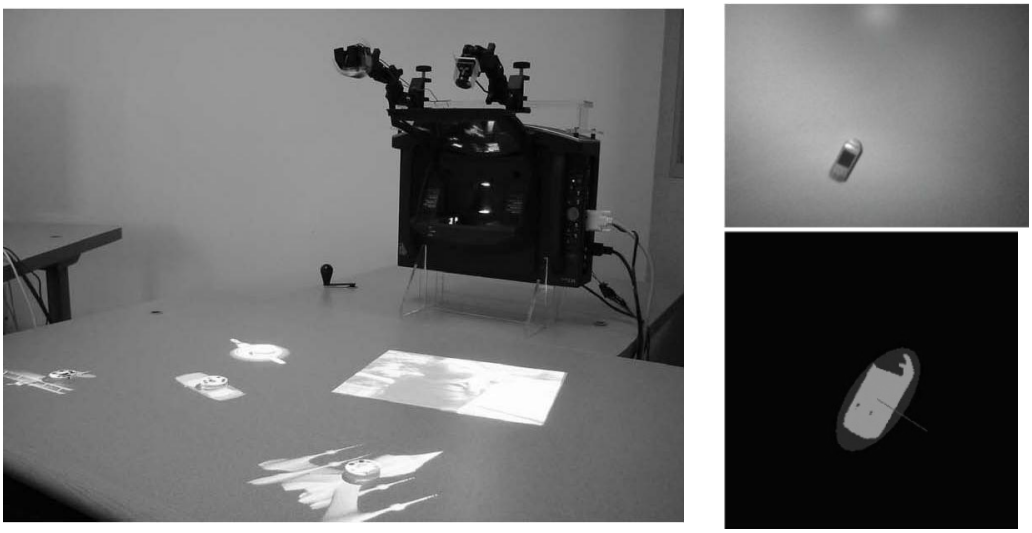

Figure 19: (Left) Steps leading to a connection between a mobile device and the BlueTable. (1) A user puts a device on the table. (2) The table's vision system detects a phone-like object on its surface and broadcasts Bluetooth signals to nearby devices instructing them to blink a certain way. (3) The device blinks on the predetermined pattern and gets detected/seen by the table. (4) A connection is established and a default interface "spills out" of the device on the table. (5) After that, the user can interact with the presented interface. (Right) Mockup of a complex user interface (calendar application) that spills out of a mobile device onto the BlueTable's surface.

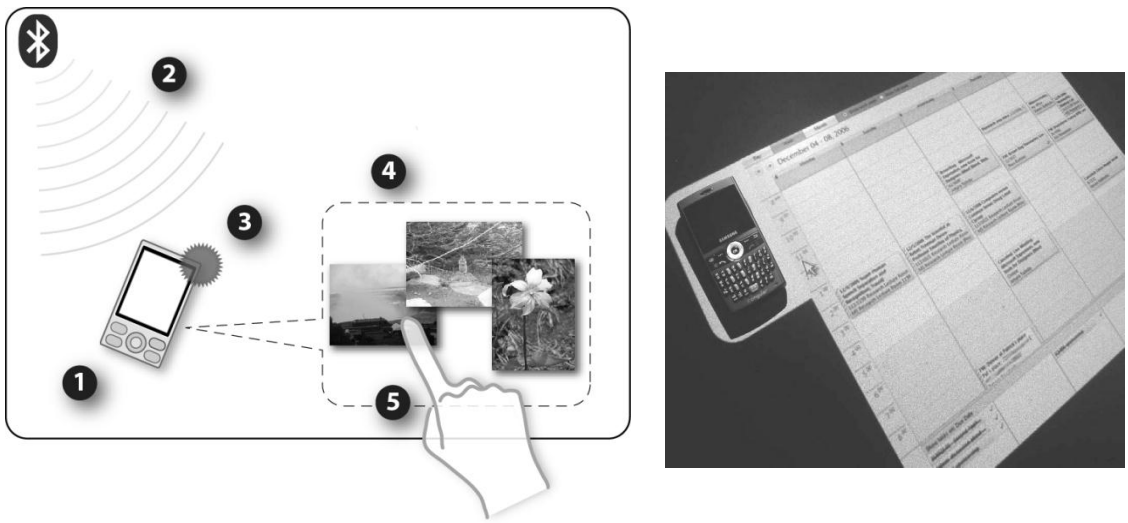

Figure 20: Our different connection mechanisms and the proxemic distances at which they can be performed.

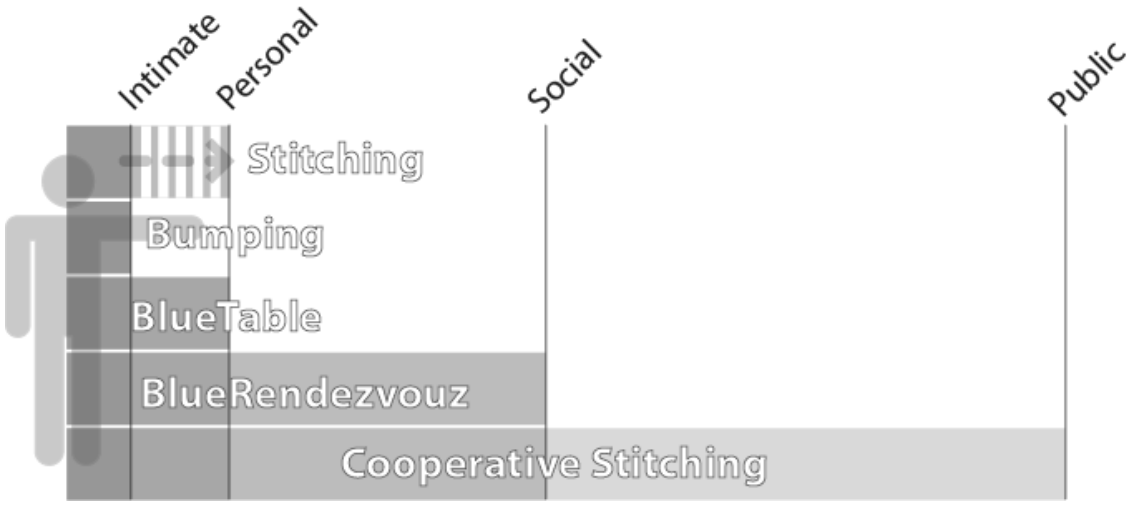

Figure 21: Mockup diagram of a stitching solution that follows a mailbox metaphor.

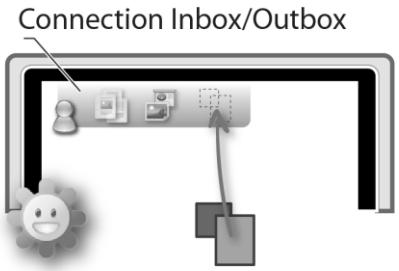

Figure 22: Mockup of an action screen to specify geometry information. The combination of keys pressed may indicate spatial operations as well as the type of function. The system does not sense the relative spatial location of the devices: users hit the arrow keys to specify this.